IMPROVING INTERNET SECURITY VIA LARGE-SCALE PASSIVE AND ACTIVE DNS MONITORING

A Thesis
Presented to
The Academic Faculty

by

Emmanouil K. Antonakakis

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Computer Science

Georgia Institute of Technology
August 2012

IMPROVING INTERNET SECURITY VIA LARGE-SCALE PASSIVE AND ACTIVE DNS MONITORING

Approved by:

Professor Wenke Lee, Advisor
School of Computer Science
Georgia Institute of Technology

Associate Professor Fabian Monrose
Department of Computer Science
University of North Carolina at Chapel Hill

Professor Mustaque Ahamad
School of Computer Science
Georgia Institute of Technology

Professor Nick Feamster
School of Computer Science
Georgia Institute of Technology

Assistant Professor Patrick Gerard Traynor
School of Computer Science
Georgia Institute of Technology

Date Approved: 17 May 2012

To Tom

for giving me a chance.

ACKNOWLEDGEMENTS

As in most aspects of life, with teamwork you can accomplish more and greater things. Throughout my research journey I was fortunate enough to interact and cooperate with a set of exceptional people that helped me mentally and physically to deliver the research in this thesis.

My advisor, Wenke Lee, guided me through all the difficulties of this journey. Reaching towards the end of this journey, I feel under his guidance I became a better person in many aspects of my life, and for that I cannot thank him enough.

My closest collaborator, Roberto Perdisci, helped me in many ways to deliver the research in this thesis. When someone sacrifices time away from his family and his son Marco, to help complete a research project, there is nothing you can say or do to thank him enough — thanks Roberto!

Of course David Dagon. David was the one that introduced me to the research area of DNS security. He taught me the basic principals of conducting research that at the same time provide merit both to the academic and operational community. His ideas still influence my way of thinking and I cannot thank him enough for this.

My academic family at the Georgia Tech Information Security Center (GTISC) clearly played a key role in my research work. My committee members, Nick Feamster, Mustaque Ahamad, Patrick Traynor and Fabian Monrose (from UNC), all provided thoughtful guidance throughout the lifecycle of this research.

Many current and former students in GTISC and folks outside of GTISC that collaborated with me on this work including Nikolaos Vasiloglou, Xiapu (Daniel) Luo, Yacin Nadji, Saeed Abu-Nimeh, Christopher P. Lee, Kevin Day, and Justin Bellmor.

Without access to real world data to validate the algorithms in this thesis the overall contribution would significantly suffer. A number of people in Damballa helped me significantly to obtain the necessary data for my research. To that extent I would like to thank my supervisor at Damballa Gunter Ollmann and a set of exceptional engineers that got their hands dirty to help me out with my research; namely Marshall T. Vandegrift, Eric Davidson, and Adam Kauffman.

The second group of people that actively helped me with data from the DNS operational community are the people from the Security Information Exchange (SIE). Without the assistance from Eric Ziegast, Paul Vixie and the exceptional Robert Edmonds parts of the research included in this thesis would not be possible.

Even more students and affiliates of GTISC including Martim Carbone, Junjie Zhang, Monirul Sharif, Kapil Singh, Long Lu, Paul Royal, Artem Dinaburg, Claudio Mazzariello, Andrea Lanzi, Igino Corona, Yizheng Chen and Michael Lee were always happy to provide feedback, advice, proofreading, and any other help needed in the lab. Also, Mary Claire Thompson and Alfreda Barrow always managed to keep GTISC running smoothly. Thanks to everyone at GTISC for providing an environment that made this work possible.

My family in Greece had to pay the largest price of my studies abroad. They had to suffer many years without me physically being close to them. This thesis is the minimal payment due for the times spent away from them and the family events missed while conducting this research. Stelios, Nick, Chris, Roula, Vicky and Loula helped me many times over my stay in Atlanta and, finally, Christina was there for me whenever I needed her to support me in any way possible throughout this long journey.

Thank you everyone, without you this research contribution would not be possible.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

The Domain Name System (DNS) is a critical component of the Internet. DNS provides the ability to map human-readable and memorable domain names to machine-level IP addresses and other records. These mappings lie at the heart of the Internet's success and are essential for the majority of core Internet applications and protocols.

The critical nature of DNS means that it is often the target of abuse. Cyber-criminals rely heavily upon the reliability and scalability of the DNS protocol to serve as an agile platform for their illicit operations. For example, modern malware and Internet fraud techniques rely upon DNS to locate their remote command-and-control (C&C) servers through which new commands from the attacker are issued, serve as exfiltration points for information stolen from the victims' computers, and to manage subsequent updates to their malicious toolset.

The research described in this thesis scientifically addresses problems in the area of DNS-based detection of illicit operations. In detail, this research studies new methods to quantify and track dynamically changing reputations for DNS based on passive network measurements. The research also investigates methods for the creation of early warning systems for DNS. These early warning systems enables the research community to identify emerging threats (e.g., new botnets and malware infections) across the DNS hierarchy in a timelier manner.

CHAPTER I

INTRODUCTION

The Domain Name System (DNS) [55, 56] maps domain names to IP addresses, and provides an important service to Internet applications. DNS is one of the core protocols in the Internet, which is often used to provide an agile and reliable infrastructure for Internet-based applications (i.e., e-commerce, on-line education etc.).

Internet-scale attacks often use DNS as well because they are essentially Internet-scale malicious applications. For example, *spyware* uses anonymously registered domains to exfiltrate private information to *drop sites*. Disposable domains are used by *adware* to host malicious or false advertising content. *Botnets* make agile use of short-lived domains to evasively move their command-and-control (C&C) infrastructure [19, 38, 39, 104]. Fast-flux networks [43, 70, 90] rapidly change DNS records to evade blacklists and resist takedowns [84, 89, 90].

To date, the security community mainly fights DNS abuse using statically compiled domain blacklists. The effectiveness of such static domain blacklists is increasingly limited, due to the overwhelming number of new domain names appearing on the Internet every day. DNS abusers frequently employ domain-fluxing techniques to run their malicious activities, thus making it difficult to keep domain blacklists up-to-date.

The security community is in need of methods that would enable DNS operators to defend against DNS abuse. Such methods should be able to provide both *situational awareness*, and the ability to *proactively detect and respond to* DNS-based Internet threats.

Improving the situational awareness of DNS resolutions would require us to quantify the notion of *dynamically changing* reputation for domain names. Law enforcement agencies have been using reputation-based methods for years, in order to predict crime events in certain areas over certain periods of the year. Their prediction abilities are based on historical observations of reports recorded by law enforcement. It is unlikely (in the near future) that the law enforcement officers would have to deal with many criminal incidents in historically safe parts of the city. In the same sense, there are areas notorious for their criminal activity.

By the systematic passive analysis of historic data on crime, law enforcement officers can more accurately forecast criminal activities. This clearly enables them to better organize their work force (i.e., frequency and schedule of patrols) but also the experience level of the law enforcement officers that will deal with potentially high-risk situations.

DNS reputation is directly linked with the historic evidence that operators can attribute to a domain name. For example, a domain name should have a low DNS reputation if it is pointing to a network for which we have proof that historically has been associated with certain levels of DNS abuse and there is no homogeneity in the domain names that can be associated with hosts in the same network. On the contrary, if a domain name is pointing to a network for which we have no evidence of historic DNS abuse and the domain names (in the same network) have a lot of consistency in their DNS structure (i.e., few effective second level domains etc.) it is very likely that it is benign.

The properties of historically related domain names in a network are very important to DNS reputation. Adversaries' key goal is the evasion of statically compiled DNS blacklists. Therefore, it is very likely that they will structure their DNS infrastructure in such a way that blacklisting one domain name (or the effective second level domain) will not affect the rest of the domain names used by them.

On the other hand, legitimate Internet applications desire a fairly stable DNS infrastructure. The main reason behind that is the quality of service and availability that legitimate professional Internet services have to provide to their users. These fundamentally different motivational observations can be used in order to systematically infer relational features for DNS.

The second goal of this thesis is to enable operators to proactively act against DNS-based Internet threats. Such early warning systems would have the potential to reduce the response time against emerging Internet threats. As in reputation systems, passive observations will be important in this effort as well.

Maybe the best examples of successful early warning systems can be found in meteorology. Historic weather patterns along with up-to-the-minute observations are used in order to obtain more accurate weather forecasts. The same analogy can be made in DNS, with one main difference: different points in the DNS hierarchy would enable the detection of different "families" of Internet threats.

In the upper layers of the DNS hierarchy, for example, top level domain (TLD) operators and authoritative name server (ANS) operators should be able to independently detect the rise of new malware related domain names. This should be made possible by the passive collection of observation from the DNS resolution requests. To achieve the quantification of such a system we study the DNS properties that can be observed in the higher levels of the DNS hierarchy as a byproduct of the malware's daily life cycle and their infection campaigns.

Domain names used for benign purposes typically have a very stable "audience" that visits them over a period of a time. It is very unlikely that a large number of new users visit benign domain names for a short period of time and then suddenly disappear [1]. However, domain names related with malware have to go through the malware infection and propagation stage. Social engineering tricks [48, 51, 53] or

[1] There could be exceptions in this claim like flash crowd type of traffic [46].

malware delivery mechanisms [14, 68, 69] may be used for the propagation of a new malware "on-the-rise".

This infection phase cannot be instant. Typically, it would be a gradual process observable over a period of time. Authoritative DNS and TLD operators are able to observe this alteration in the DNS resolution patterns. This change in the DNS resolution behavior can be used along with historic observations from malware-related domain names to enable the early warning of new malware-related domain names on the rise.

A single malware family can evade detection at the upper layer of the DNS hierarchy. Malware that employ domain name generation algorithms (DGA) to establish their command and control (C&C) communication make use of disposable domain names. Typically such disposable domain names are active for a small period of time, leaving very limited historic traces at the TLD or ANS operators. A byproduct of the normal DGA operation is the generation of Name Error responses or NXDomains, which are observable at the recursive levels of the DNS hierarchy. In order to provide a complete early warning system for malware related domain names across the DNS hierarchy, we need to be able to detect DGAs on the rise at the recursive DNS levels. As discussed previously, we make use of passive observations of NXDomains and we model new DGAs on the rise by observing properties of the generated NXDomains.

1.1 Motivation and Challenges

The effectiveness of static domain blacklists is increasingly limited. The main reason for this limitation is the overwhelming number of new domain names appearing on the Internet every day. Attackers frequently switch to different domains to run their malicious activities, thus making it difficult to keep blacklists up-to-date. Attackers now make very aggressive use of *DNS agility*, which provides a straightforward technique to evade static domain name blacklisting.

To overcome the limitations of static domain blacklists, the security community needs a new generation of DNS-based detection systems that will be able to:

- *Dynamically* assign reputation scores to domain names according to their successful DNS resolutions.
- *Dynamically* detect new malware-related domains in various parts of the DNS hierarchy.
- *Dynamically* identify the rise of "Domain Name Generation Algorithm"-based (or DGA) botnets, using properties of the DGA-based botnets communication cycle observable at the local recursive DNS level.

1.1.1 DNS Reputation

Over the years, Internet miscreants have used DNS to build malicious network infrastructures. For example, botnets and other types of malicious software make use of domain names to locate their command and control (C&C) servers and communicate with attackers, e.g., to exfiltrate stolen private information, wait for commands to perform attacks on other victim machines, etc.

In response to this malicious use of DNS, network operators use static domain blacklists to contain known malware domains to flag DNS queries originating from malware-infected machines. Such static domain name blacklists try to effectively block the communication channel between infected machines and the attackers. Unfortunately, static blacklists, which typically still are manually compiled, cannot keep pace with the level of network agility of modern Internet threats.

1.1.2 Early Detection of Malware Outbreaks

DNS reputation systems at the recursive level or at the edge of the monitored network can identify potentially malicious domain names according to their successful DNS resolutions. In order to achieve early domain name detection on new malware-related

domain names on the rise, we need to be able to classify domain names at the higher levels of the DNS hierarchy, not only at the recursive level.

Only one known malware family could potentially evade detection systems that operate at the higher levels of the DNS hierarchy. Malware that make use of domain name generation algorithms (DGAs) for connecting back to their C&Cs typically utilize multiple (different) authorities and domain names. It would require an unrealistically wide deployment of DNS sensors across multiple different DNS authorities to model the domain names that such a malware would use, at the authority level. However, at the recursive or local network level, such DGA-based malware inevitably leave network traces of non-successful DNS resolutions (NXDomains) as part of their normal C&C discovery cycle. This DNS behavior is yet to be harvested by classification systems for early detection of new DGAs "on-the-rise".

1.1.2.1 Leveraging the Higher Levels of The DNS Hierarchy

While domain reputation systems at the recursive DNS level represent an important advancement, they have limited, local capabilities in that they can detect new malicious domains only after the corresponding malicious activities have propagated to their monitored (or cooperative) networks. Domain name reputation systems classify primarily based on resource records, the domain name and its resolved IP address. By leveraging the properties of resource records, current domain reputation systems are heavily dependent on historic observations about the domain names and the IP addresses they point to.

Given the nature of emerging Internet threats, it is important that we develop global monitoring capabilities to stop impending attacks on our networks. Furthermore, there is a need for a variety of classification signals that enable early detection of malicious domain names. Such a system should not rely heavily upon a single class of detection signals (e.g., historic IP reputation). This is because evading a

detector that uses a variety of detection signals becomes significantly harder and the detector will no longer be prone to sudden changes in its main detection signals (e.g., transition from IPv4 to IPv6 reputation).

1.1.2.2 Leveraging the Unsuccessful DNS Resolutions at the Recursive Level

DGA-based botnets (i.e., Srizbi, Conficker, Murofet etc.) force the defenders to: (i) reverse-engineer the DGA algorithm; (ii) use the obtained DGA to pre-compute future candidate C&C domains; (iii) register the generated domains before the botmaster does. While the botmaster only needs to register a few domains per day to maintain control over the botnet (with high probability), a comprehensive take down effort would require security operators to pre-register a very large number of domains per day, which quickly becomes very costly.

Alternatively, security operators may seek the collaboration of domain registrars to stop anybody, including the botmaster, from registering domains produced by the DGA. Unfortunately, while this is an appealing solution that has been implemented to counter Conficker, the fact that Conficker may use 110 different TLDs made this solution fairly laborious and requires the collaboration between network operators (e.g., country-code TLD administrators) across many different national boundaries.

Unfortunately, the security community typically discovers new malware weeks, or even months after the first infection takes place [8]. To address the early detection of new emerging DGA-based botnets without requiring malware samples, we need to invent new detection systems. Such systems should be able to accurately detect machines within a network that are compromised with DGA-based bots by analyzing streams of unsuccessful DNS resolutions. Such detection systems should be able to operate at the local recursive level or at the edge of a network so operators can independently identify DGA-based compromised machines in their networks.

1.2 Contributions

Dynamic Reputation for DNS: To address the limitation of static domain name blacklists we developed Notos [7], a dynamic reputation system for DNS. Notos uses passive DNS evidence from recursive DNS servers to distinguish between benign and malicious domain names using historical learning techniques. Notos allows us to statistically correlate the two planes in DNS: the name space and the address space. The primary goal of Notos is to automatically assign a low reputation score to a domain that is involved in malicious activities, such as malware C&C, "phishing", and spam campaigns. Conversely, we want to assign a high reputation score to domains that are used for legitimate purposes.

DNS-based Malware Detection at the DNS Authority Level: The first component of the early warning system we developed is named Kopis [8]. Kopis operates in the upper layers of the DNS hierarchy and is capable of detecting malware-related domain names "on-the-rise". This early warning system can be independently deployed and operated by the top-level domain (TLD) and authoritative DNS (ANS) operators. The system enables TLD and ANS operators to detect malware-related domains from within their authority zones without the need for data from other networks or other inter-organizational coordination. The detection of such malware related domain names typically comes days or even weeks before the domains appear in public blacklists.

DNS-based Malware Detection at the DNS Recursive Level: Pleiades [9] is the second component of our early warning system against rising malware threats. In particular Pleiades is able to detect the rise of DGA botnets in a local network by statistical modeling of the unsuccessful DNS resolutions at the recursive DNS level of the monitored network. Pleiades is able to learn models from traffic generated by

already known DGA-based malware and to detect active infections in the monitored networks.

1.3 Dissertation overview

In Chapter 2, we introduce the basic terms that we will use in the thesis. We also discuss the necessary background material and we discuss the related work.

In Chapter 3, we elaborate on Notos, the dynamic reputation system for DNS. Section 3.2 describes in detail our passive DNS collection strategy and other white-list and blacklist inputs. We also describe three feature extraction modules that measure key network, zone and evidence-based features. Finally, we describe how these features are clustered and incorporated into the final reputation engine. To evaluate the output of Notos, we gathered an extensive amount of network trace data. Section 3.3 describes the data collection process, while in Section 3.4 we present the statistical feature analysis. The sensitivity of each module and the analysis on the experiment results can be found in Section 3.5.

In Chapter 4, we provide an in-depth analysis of Kopis, the early warning system for malware-related domain names that operates in the upper layers of the DNS hierarchy. In Section 4.2, we introduce Kopis, and in Section 4.3, we elaborate in detail how we compute Kopis' statistical features. Section 4.4 presents an analysis of the characteristics of the DNS traffic we used to evaluate our detection system. Additionally, we report all detection results and discoveries made possible through Kopis.

In Chapter 5, we introduce Pleiades, an early warning system capable of detecting DGAs on the rise at the recursive DNS levels. We provide an overview of Pleiades in Section 5.2. The DGA discovery process is described in Section 5.3. Section 5.4 describes the DGA classification and C&C detection processes. We elaborate on the properties of the datasets used and the way we obtained the ground truth in

Section 5.5. The experimental results are presented in Section 5.6.

We conclude the thesis in Chapter 6. Section 6.1 summaries the contributions. In Section 6.2 we discuss the operational aspects of all systems presented in this thesis along with the possible evasion techniques against them. Next, in Section 6.3 we discuss the open research problems that are related with the research presented in this thesis. We conclude in Section 6.4 with the closing remarks.

CHAPTER II

BACKGROUND AND PREVIOUS WORK

2.1 Background

2.1.1 The Domain Name System

In the Domain Name System (DNS), domain names are composed of labels, separated by periods, which correspond to namespaces in a hierarchical tree structure. Each domain name is a node, and the bottom-up concatenation of node names creates a fully qualified domain name. A zone is a collection of such nodes, constituting a separate tree structure, with the zone's *start of authority*, or `SOA`, at the apex. The contents of the zone (either mappings of labels to hosts, or further downward delegation), is available from DNS *authority name servers* (or ANS). In DNS nomenclature, these authority servers are sometimes called the SOA or ANS.

A *recursive DNS resolver* (or RDNS) is what one normally thinks of as a "DNS server". Such resolvers accept queries from users, understand the zone hierarchy system, and properly implement the various rules and RFCs to obtain and cache answers.

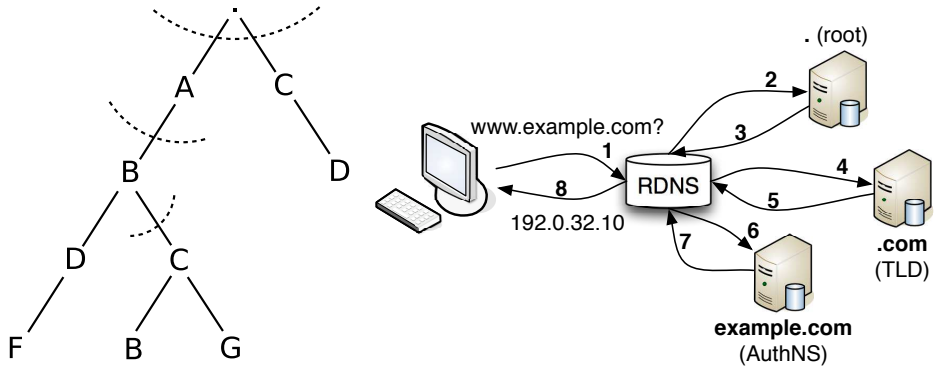
DNS initiators on host machines are called *stub resolvers*. They typically don't interact with the zone hierarchy, and with a few exceptions, don't cache answers. Instead, they implement enough DNS logic to pose basic queries to recursive DNS servers.

The domain name space is structured like a tree. A domain name identifies a node in the tree. For example, the domain name `F.D.B.A.` identifies the path from the root "." to a node `F` in the tree (see Figure 1(a)). The set of resource information associated with a particular name is composed of resource records (RRs) [55, 56].

The depth of a node in the tree is sometimes referred to as *domain level*. For example, `A.` is a top-level domain (TLD), `B.A.` is a second-level domain (2LD), `D.B.A.` is a third-level domain (3LD), and so on.

The information related to the domain name space is stored in a distributed *domain name database*. The domain name database is partitioned by "cuts" made in the name space between adjacent nodes. After all cuts are made, each group of connected nodes represent a separate *zone* [55]. Each zone has at least one node, and hence a domain name, for which it is *authoritative*. For each zone, a node which is closer to the root than any other node in the zone can be identified. The name of this node is often used to identify the zone.

The RRs of the nodes in a given zone are served by one or more *authoritative* name servers (ANS or AuthNSs). AuthNSs that have complete knowledge about a zone (i.e., they store the RRs for all the nodes related to the zone in question in its zone files) are said to have *authority* over that zone [55, 56]. AuthNSs will typically support one or more zones, and can delegate the authority over part of a (sub-)zone to other AuthNSs.



(a) DNS Tree (b) Domain Resolution

Figure 1: Example of DNS tree and domain resolution process.

DNS queries are usually initiated by a *stub resolver* on a user's machine, which

relies on a *recursive DNS resolver* (RDNS) for obtaining a set of RRs owned by a given domain name. The RDNS is responsible for directly contacting the AuthNSs on behalf of the stub resolver to obtain the requested information, and return it to the stub resolver. The RDNS is also responsible for caching the obtained information no longer than a certain period of time, called the *Time To Live* (TTL), so that if the same or another stub resolver queries again for the same information within the TTL time window, the RDNS will not need to contact the authoritative name servers (thus improving efficiency). Figure 1(b) enumerates the steps involved in a typical query resolution process, assuming an empty cache.

Briefly, in step one the stub resolver requests a resolution for `www.example.com`. Assuming the RDNS server has no cached information whatsoever, it starts the iterative process from step two. In this step it requests from the *root servers* a referral for the `www.example.com`. The *root servers* (in step three) provides back the delegation for *www.example.com* pointing the recursive to the *.com TLD servers*. Following the same iteration (steps four and five) the *.com TLD servers* refers the RDNS server to the *authoritative DNS server (AuthNS)* for the zone `example.com`.

The final iterative steps will happen in steps six and seven where the RDNS receives the DNS resolution for the domain name `www.example.com`. Finally, in step eight the RDNS server provides back to the stub the DNS resolution for the domain name `www.example.com`. From the AuthNS point of view we call as the `left hand` DNS the IP address of the RDNS looking up the domain name `www.example.com`. In the same sense, we will call as the `right hand` DNS the IP address that the domain name `www.example.com` points to — in this case the IP address will be `192.0.32.10`. In the interest of space, we refer the reader to [55,56,93] for more detailed information on the DNS and special cases of the query resolution process.

2.2 Previous Work

2.2.1 Passive DNS Replication

By aggregating all **unique**, successfully resolved `A-type` DNS answers at the recursive level, one can build a passive DNS database. This passive DNS (pDNS) database is effectively the DNS fingerprint of the monitored network and typically contains unique `A-type` resource records (RRs) that were part of monitored DNS answers. A typical RR for the domain name `example.com` has the following format: {`example.com. 78366 IN A 192.0.2.10`}, which lists the domain name, TTL, class, type, and rdata. For simplicity, we will refer to an RR as just a tuple of the domain name and IP address.

Passive DNS data collection was first proposed by Florian Weimer [95]. His system was among the first that appeared in the DNS community with its primary purpose being the conversion of historic DNS traffic into an easily accessible format. Zdrnja et al. [106] with their work in "Passive Monitoring of DNS Anomalies" discuss how pDNS data can be used for gathering security information from domain names. Although they acknowledge the possibility of creating a DNS reputation system based on passive DNS measurement, they do not quantify a reputation function.

Notos [7] uses the idea of building passive DNS information only as a seed for computing statistical DNS properties for each successful DNS resolution. The analysis of these statistical properties is the basic building block for our dynamic domain name reputation function. Plonka et al. [63] introduced Treetop, a scalable way to manage a growing collection of passive DNS data and at the same time correlate zone and network properties. Their cluster zones are based on different classes of networks (class A, class B and class C). Treetop differentiates DNS traffic based on whether it complies with various DNS RFCs and based on the resolution result. Plonka's proposed method, despite being novel and highly efficient, offers limited DNS security information and cannot assign reputation scores to records.

2.2.2 TLD Measurements

To the best of our knowledge, Wessels et al. [96] were the first to analyze DNS query data as seen from the upper DNS hierarchy. The authors focused on examining the DNS caching behavior of recursive DNS servers from the point of view of AuthNS and TLD servers, and how different implementations of caching systems may affect the performance of the DNS.

Recently, Hao et al. [40] released a report on DNS lookup patterns measured from the `.com` TLD servers. Their preliminary analysis shows that the resolution patterns for malicious domain names are sometimes different from those observed for legitimate domains. While [40] only reports some preliminary measurement results and does not discuss how the findings may be leveraged for detection purposes, it does hint that a malware detection system may be built around TLD-level DNS queries. We designed Kopis [8] to do just that, namely monitor query streams at the upper DNS hierarchy and be able to detect previously unknown malware domains.

2.2.3 Malware Propagation

Several studies provide deep understanding behind the properties of malware propagation and botnet's lifetime [20, 81, 94]. An interesting observation among all these research efforts is the inherent diversity of the botnet's infected population. Collins et al. [16] introduced and quantified the notion of "network uncleanliness" from the temporal and spatial network point of view, showing that it is very probable to have a large number of infected bots in the same network over an epoch. They also discuss that this could be a direct effect of the network policy enforced at the edge. Kopis [8] directly uses the intuition behind these past research efforts in the *requester diversity* and *requester profile* statistical feature families.

2.2.4 DNS and IP Address Space Reputation

Several research studies, e.g., Sinha et al. [78] have studied the effectiveness of IP blacklists. Zhang, et al. [109] showed that the hit rate of highly predictable blacklists (HBLs) decreases significantly over a period of time. Notos [7] addresses the dynamic DNS blacklisting problem that makes it significantly different from the highly predictable blacklists. Importantly, Notos [7] does not aim to create IP blacklists. By using properties of the DNS protocol, Notos [7] can rank a domain name as potentially malicious or not. Garera et al. [34] discussed "phishing" detection predominately using properties of the URL and not statistical observations about the domains or the IP address. The statistical features used by Holz et al. [43] to detect fast flux networks are similar to the ones we used in our work, however, Notos [7] utilizes a more complete collection of network statistical features and is not limited to fast flux networks detection.

Researchers have attempted to use unique characteristics of malicious networks to detect sources of malicious activity. Anderson et al. [6] proposed Spamscatter as the first system to identify and characterize spamming infrastructure by utilizing layer 7 analysis (i.e., web sites and images in spam). Hao et al. [73] proposed SNARE, a spatio-temporal reputation engine for detecting spam messages with very high accuracy and low false positive rates. The SNARE reputation engine is the first work that utilized statistical network-based features to harvest information for spam detection. Notos [7] is complementary to SNARE and Spamscatter, and extends both to not only detect spam, but also identify other malicious activity such as phishing and malware hosting. Qian et al. [105] present their work on spam detection using network-based clustering. In this work, they show that network-based clusters can increase the accuracy of spam-oriented blacklists. Our work is more general, since we try to identify various kinds of malicious domain names. Nevertheless, both works leverage network-based clustering for identifying malicious activities.

Felegyhazi et al. [27] proposed a DNS reputation blacklisting methodology based on `WHOIS` observations. Our systems does not use `WHOIS` information making our approaches complementary by design. Sato et al. [75] proposed a way to extend current blacklists by observing the co-occurrence of IP address information. Notos [7] is a more generic approach than the proposed system by Sato and is not limited to botnet related domain name detection. Finally, Notos [7] builds the reputation function mainly based upon passive information from DNS traffic observed in real networks — not traffic observed from honeypots.

2.2.5 NXDomain Analysis

Dynamic domain generation has been used by malware to evade detection and complicate mitigation, e.g., Bobax, Kraken, Torpig, Srizbi, and Conficker [65]. To uncover the underlying domain generation algorithm (DGA), researchers often need to reverse engineer the bot binary. Such a task can be time consuming and requires advanced reverse engineering skills [52].

The infamous Conficker worm is one of the most aggressive pieces of malware with respect to domain name generation. The "C" variant of the worm generated 50,000 domains per day. However, Conficker-C only queried 500 of these domains every 24 hours. In older variants of the worm, A and B, the worm cycled through the list of domains every three and two hours, respectively. In Conficker-C, the length of the generated domains was between four and ten characters, and the domains were distributed across 110 TLDs [66].

Stone-Gross et al. [84] were the first to introduce domain fluxing. In the past, malware used IP fast-fluxing, where a single domain name pointed to several IP addresses to avoid being taken down easily. However, in domain fluxing malware uses a domain generation algorithm to generate several domain names, and then attempt to communicate with a subset of them. The authors also analyzed Torpig's

DGA and found that the bot utilizes Twitter's API. Specifically, it used the second character of the most popular Twitter search and generated a new domain every day. It was updated to use the second character of the 5^{th} most popular Twitter search. Srizbi [99] is another example of a bot that utilizes a DGA by using unique magic number. Researchers identified several unique magic numbers from multiple copies of the bot. The magic number is XOR'ed with the current date and a different set of domains is generated. Only the characters "q w e r t y u i o p a s d f" are used in the generated domain names.

Yadav et. al. proposed a technique to identify botnets by finding randomly generated domain names [102], and improvements that also include NXDomains and temporal correlation [101]. They evaluated their approaches by automatically detecting Conficker botnets in an offline dataset from a Tier-1 ISP in South Asia in the first research study [102], and both the ISP dataset and a university's DNS logs in the second [101].

Villamarin-Salomon and Brustoloni [92] compared two approaches to identify botnet C&Cs. In their first approach, they identified domains with high query rates or domains that were temporally correlated. They used Chebyshev's inequality and Mahalanobis distance to identify anomalous domains. In their second approach, they analyzed recurring "dynamic" DNS replies with NXDomain responses. Their experiments showed that the first approach was ineffective, as several legitimate services use DNS with short time-to-live (TTL) values. However, their second approach yielded better detection and identified suspicious C&C domains.

Pleiades [9] differs from the approaches described above in the following ways. **(A)** Our work models five different types of bot families including Conficker, Murofet, Sinowal, and Bobax. **(B)** We model these bot families using two clustering techniques. The first utilizes the distribution of the characters and 2-grams in the domain name. The second relies on historical data that shows the relationship between hosts and

domain names. **(C)** We build a classification model to predict the maliciousness of domains that deviate from the two clustering techniques.

Unlike previous work, Pleiades [9] does not require active probing to maintain a fresh list of legitimate domains. Our approach does not rely on external reputation databases (e.g., DNSBLs); instead, it only requires access to local DNS query streams to identify new clusters of DGA NXDomains. Not only does our approach identify new DGAs, but it also builds models for these DGAs to classify hosts that will generate similar NXDomains in the future. Furthermore, among the list of identified domains in the DGAs, our approach pinpoints the C&C domains.

CHAPTER III

ESTABLISHING DNS REPUTATION

3.1 Motivation

The Domain Name System (DNS) [55, 56] maps domain names to IP addresses, and provides a core service to applications on the Internet. DNS is also used in network security to distribute IP reputation information, e.g., in the form of DNS-based Block Lists (DNSBLs) used to filter spam [24, 67] or block malicious web pages [59, 91].

Internet-scale attacks often use DNS as well because they are essentially Internet-scale malicious applications. For example, *spyware* uses anonymously registered domains to ex-filtrate private information to *drop sites*. Disposable domains are used by *adware* to host malicious or false advertising content. *Botnets* make agile use of short-lived domains to evasively move their command-and-control (C&C) infrastructure. Fast-flux networks rapidly change DNS records to evade blacklists and resist take downs [90]. In an attempt to evade domain name blacklisting, attackers now make very aggressive use of *DNS agility*. The most common example of an *agile* malicious resource is a fast-flux network, but DNS agility takes many other forms including disposable domains (e.g., tens of thousands of randomly generated domain names used for spam or botnet C&C), domains with dozens of `A` records or `NS` records (in excess of levels recommended by RFCs, in order to resist takedowns), or domains used for only a few hours of a botnet's lifetime. Perhaps the best example is the Conficker.C *worm* [60]. After Conficker.C infects a machine, it will try to contact its C&C server, chosen at random from a list of 50,000 possible domain names created every day. Clearly, the goal of Conficker.C was to frustrate blacklist maintenance and take-down efforts. Other malware that abuse DNS include Sinowal (a.k.a. Torpig) [37],

Kraken [72], and Srizbi [74]. The aggressive use of newly registered domain names is seen in other contexts, such as spam campaigns and malicious flux networks [70, 90]. This strategy delays takedowns, degrades the effectiveness of blacklists, and pollutes the Internet's name space with unwanted, discarded domains.

In this Chapter we study the problem of *dynamically* assigning *reputation* scores to new, unknown domains. Our main goal is to automatically assign a low reputation score to a domain that is involved in malicious activities, such as malware spreading, phishing, and spam campaigns. Conversely, we want to assign a high reputation score to domains that are used for legitimate purposes. The reputation scores enable *dynamic* domain name blacklists to counter cyber attacks much more effectively. For example, with static blacklisting, by the time one has sufficient evidence to put a domain on a blacklist, it typically has been involved in malicious activities for a significant period of time. With dynamic blacklisting our goal is to decide, even for a new domain, whether it is likely used for malicious purposes. To this end, we propose Notos, a system that dynamically assigns reputation scores to domain names. Our work is based on the observation that agile malicious uses of DNS have unique characteristics, and can be distinguished from legitimate, professionally provisioned DNS services. In short, network resources used for malicious and fraudulent activities inevitably have distinct *network characteristics* because of their need to evade security countermeasures. By identifying and measuring these features, Notos can assign appropriate reputation scores.

Notos uses historical DNS information collected passively from multiple recursive DNS resolvers distributed across the Internet to build a model of how network resources are allocated and operated for legitimate, professionally run Internet services. Notos also uses information about malicious domain names and IP addresses obtained from sources such as spam-traps, honeynets, and malware analysis services to build a model of how network resources are typically allocated by Internet miscreants. With

these models, Notos can assign reputation scores to new, previously unseen domain names, therefore enabling dynamic blacklisting of unknown malicious domain names and IP addresses. While previous work on dynamic reputation systems mainly focused on IP reputation [6, 73, 78, 109]. To the best of our knowledge, Notos is the first comprehensive *dynamic* reputation system around domain names.

3.1.1 Contributions

To summarize, our main contributions are as follows. We designed Notos, a dynamic, comprehensive reputation system for DNS that outputs reputation scores for domains. We constructed *network* and *zone* features that capture the characteristics of resource provisioning, usages, and management of domains. These features enable Notos to learn models of how legitimate and malicious domains are operated, and compute accurate reputation scores for new domains.

We implemented a proof-of-concept version of our system, and deployed it in a large ISP's DNS network in Atlanta, GA and San Jose, CA, USA, where we observed DNS traffic from 1.4 million users. We also used passive DNS data from the Security Information Exchange (SIE) project [17].

This extensive *real-world* evaluation shows Notos can correctly classify new domains with a low false positive rate (0.38%) and high true positive rate (96.8%). Notos can detect and assign a low reputation score to malware- and spam-related domain names several days or even weeks before they appear on public blacklists.

3.2 A Dynamic Reputation System for DNS

The goal of the Notos reputation system is to dynamically assign reputation scores to domain names. Given a domain name d, we want to assign a low reputation score if d is involved in malicious activities (e.g., if it has been involved with botnet C&C servers, spam campaigns, malware propagation, etc.). On the other hand, we want to assign a high reputation score if d is associated with legitimate Internet services.

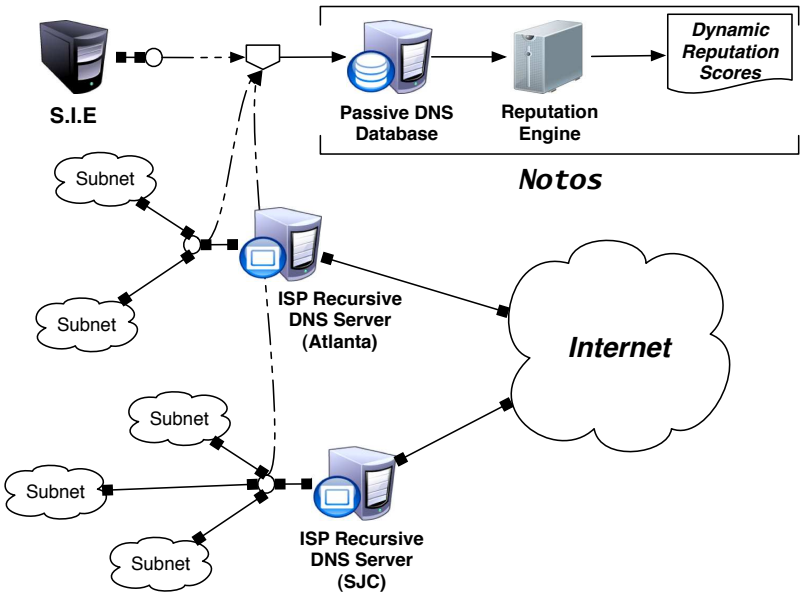


Figure 2: High level overview of Notos.

Notos' main source of information is a passive DNS (pDNS) database, which contains historical information about domain names and their resolved IPs. Our pDNS database is constantly updated using real-world DNS traffic from multiple geographically diverse locations as shown in Figure 2. We collect DNS traffic from two ISP recursive DNS servers (RDNS) located in Atlanta and San Jose. The ISP nodes witness 30,000 DNS queries/second during peak hours. We also collect DNS traffic through the Security Information Exchange (SIE) [17], which aggregates DNS traffic received by a large number of RDNS servers from authoritative name servers across North America and Europe. In total, the SIE project processes approximately 200 Mbit/s of DNS messages, several times the total volume of DNS traffic in a single US ISP.

Another source of information we use is a list of known malicious domains. For example, we run known malware samples in a controlled environment and we classify as suspicious all the domains contacted by malware samples that do not match a pre-compiled white list. In addition, we extract suspicious domain names from spam

emails collected using a large spam-trap. Again, we discard the domains that match our whitelist and consider the rest as potentially malicious. Furthermore, we collect a large list of popular, legitimate domains from `alexa.com` (we discuss our data collection and analysis in more details in Section 3.3). The set of known malicious and legitimate domains represents our *knowledge base*, and is used to train our reputation engine, as we discuss in Section 3.3.

Intuitively, a domain name d can be considered suspicious when there is evidence that d or its IP addresses are (or were in previous months) associated with known malicious activities. The more evidence of "bad associations" we can find about d, the lower the reputation score we will assign to it. On the other hand, if there is evidence that d is (or was in the past) associated with legitimate, professionally run Internet services, we will assign it a higher reputation score.

3.2.1 System Overview

Before describing the internals of our reputation system, we introduce some basic terminology. A domain name d consists of a set of substrings or labels separated by a period; the rightmost label is called the *top-level* domain, or TLD. The *second-level* domain (2LD) represents the two rightmost labels separated by a period; the *third-level* domain (3LD) analogously contains the three rightmost labels, and so on. As an example, given the domain name d="a.b.example.com", $TLD(d)$="com", $2LD(d)$="example.com", and $3LD(d)$="b.example.com".

Let s be a domain name (e.g., s="example.com"). We define $Zone(s)$ as the set of domains that include s and all domain names that end with a period followed by s (e.g., domains ending in ".example.com").

Let $D = \{d_1, d_2, ..., d_m\}$ be a set of domain names. We call $A(D)$ the set of IP addresses ever pointed to by any domain name $d \in D$.

Given an IP address a, we define $BGP(a)$ to be the set of all IPs within the BGP

prefix of a, and $AS(a)$ as the set of IPs located in the autonomous system in which a resides. In addition, we can extend these functions to take as input a set of IPs: given IP set $A = a_1, a_2, ..., a_N$, $BGP(A) = \bigcup_{k=1..N} BGP(a_k)$; $AS(a)$ is similarly extended.

To assign a reputation score to a domain name d we proceed as follows. First, we consider the most current set $A_c(d) = \{a_i\}_{i=1..m}$ of IP addresses to which d points. Then, we query our pDNS database to retrieve the following information. **Related Historic IPs (RHIPs):** RHIPs consist of the union of $A(d)$, $A(Zone(3LD(d)))$, and $A(Zone(2LD(d)))$. In order to simplify the notation we will refer to $A(Zone(3LD(d)))$ and $A(Zone(2LD(d)))$ as $A_{3LD}(d)$ and $A_{2LD}(d)$, respectively. **Related Historic Domains (RHDNs):** RHDNs comprise the entire set of domain names that ever resolved to an IP address $a \in AS(A(d))$. In other words, RHDNs contain all the domains d_i for which $A(d_i) \cap AS(A(d)) \neq \emptyset$.

After extracting the above information from our pDNS database, we measure a number of statistical features. Specifically, for each domain d we extract three groups of features, as shown in Figure 3:

- **Network-based features:** The first group of statistical features is extracted from the set of RHIPs. We measure quantities such as the total number of IPs historically associated with d, the diversity of their geographical location, the number of distinct autonomous systems (ASs) in which they reside, etc.

- **Zone-based features:** The second group of features we extract are those from the RHDNs set. We measure the average length of domain names in RHDNs, the number of distinct TLDs, the occurrence frequency of different characters, etc.

- **Evidence-based features:** The last set of features includes the measurement of quantities such as the number of distinct malware samples that contacted the domain d, the number of malware samples that connected to any of the IPs

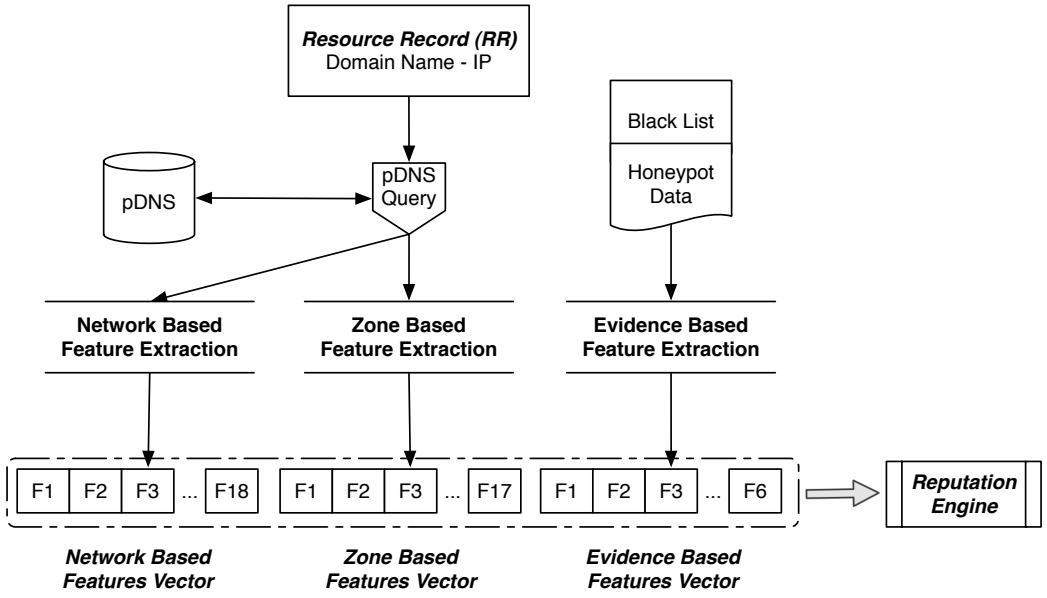


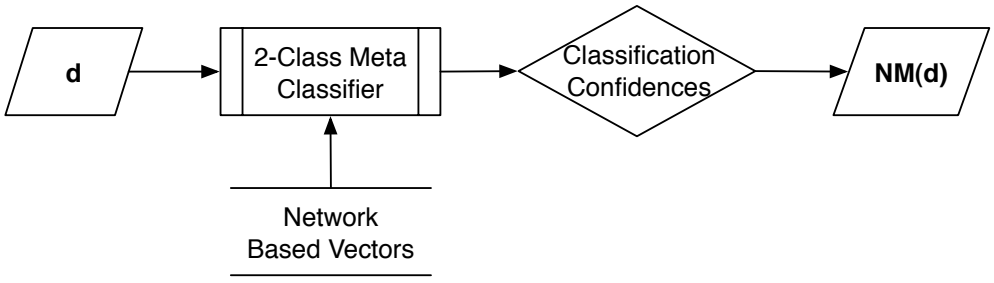
Figure 3: Computing network-based, zone-based, evidence-based features.

pointed by d, etc.

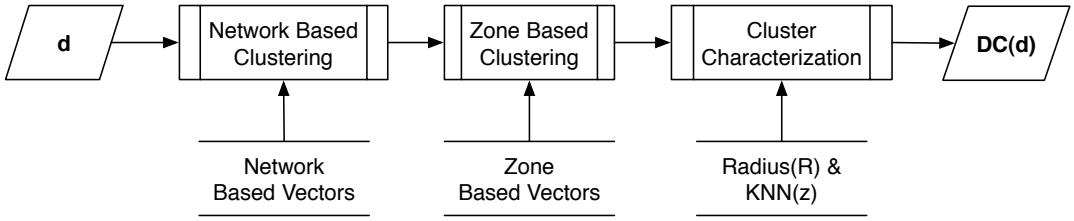
Once extracted, these statistical features are fed to the reputation engine. Notos' reputation engine operates in two modes: an off-line "training" mode and an on-line "classification" mode. During the off-line mode, Notos *trains* the reputation engine using the information gathered in our *knowledge base*, namely the set of known malicious and legitimate domain names and their related IP addresses. Afterwards, during the on-line mode, for each new domain d, Notos queries the trained reputation engine to compute a reputation score for d (see Figure 5). We now explain the details about the statistical features we measure, and how the reputation engine uses them during the off-line and on-line modes to compute a domain names' reputation score.

3.2.2 Statistical Features

In this section we identify key statistical features and the intuition behind their selection.



(a)



(b)

Figure 4: (a) Network profile modeling in Notos. (b) Network and zone based clustering in Notos.

3.2.2.1 Network-based Features

Given a domain d we extract a number of statistical features from the set RHIPs of d, as mentioned in Section 3.2.1. Our network-based features describe how the operators who *own* d and the IPs that domain d points to, allocate their network resources. Internet miscreants often abuse DNS to operate their malicious networks with a high level of *agility*. Namely, the domain names and IPs that are used for malicious purposes are often short-lived and are characterized by a high *churn* rate. This agility avoids simple blacklisting or removals by law enforcement. In order to measure the level of agility of a domain name d, we extract eighteen statistical features that describe d's *network profile*. Our network features fall into the following three groups:

- *BGP features.* This subset consists of a total of nine features. We measure the number of distinct BGP prefixes related to $BGP(A(d))$, the number of countries

in which these BGP prefixes reside, and the number of organizations that *own* these BGP prefixes; the number of distinct IP addresses in the sets $A_{3LD}(d)$ and $A_{2LD}(d)$; the number of distinct BGP prefixes related to $BGP(A_{3LD}(d))$ and $BGP(A_{2LD}(d))$, and the number of countries in which these two sets of prefixes reside.

- *AS features.* This subset consists of three features, namely the number of distinct autonomous systems related to $AS(A(d))$, $AS(A_{3LD}(d))$, and $AS(A_{2LD}(d))$.

- *Registration features.* This subset consists of six features. We measure the number of distinct registrars associated with the IPs in the $A(d)$ set; the diversity in the registration dates related to the IPs in $A(d)$; the number of distinct registrars associated with the IPs in the $A_{3LD}(d)$ and $A_{2LD}(d)$ sets; and the diversity in the registration dates for the IPs in $A_{3LD}(d)$ and $A_{2LD}(d)$.

While most legitimate, professionally run Internet services have a very stable network *profile*, which is reflected into low values of the network features described above, the profiles of malicious networks (e.g., fast-flux networks) usually change relatively frequently, thus causing their network features to be assigned higher values. We expect a domain name d from a legitimate zone to exhibit a small values in its AS features, mainly because the IPs in the RHIPs should belong to the same organization or a small number of different organizations. On the other hand, if a domain name d participates in malicious activities (i.e., botnet activities, flux networks), then it could reside in a large number of different networks. The list of IPs in the RHIPs that correspond to the malicious domain name will produce AS features with higher values. In the same sense, we measure that homogeneity of the registration information for benign domains. Legitimate domains are typically linked to address space owned by organizations that acquire and announce network blocks in some order. This means that the registration-feature values for a legitimate domain name d that

owned by the same organizations will produce a list of IPs in the RHIPs that will have small registration feature values. If this set of IPs exhibits high registration feature values, it means that they very likely reside in different registrars and were registered on different dates. Such registration-feature properties are typically linked with fraudulent domains.

3.2.2.2 Zone-based Features

The network-based features measure a number of characteristics of IP addresses historically related to a given domain name d. On the other hand, the zone-based features measure the characteristics of domain names historically associated with d. The intuition behind the zone-based features is that while legitimate Internet services may be associated with many different domain names, these domain names usually have strong similarities. For example, `google.com`, `googlesyndication.com`, `googlewave.com`, etc., are all related to Internet services provided by Google, and contain the string "google" in their name. On the other hand, malicious domain names related to the same spam campaign, for example, often look randomly generated and share few common characteristics. Therefore, our zone-based features aim to measure the level of *diversity* across the domain names in the RHDNs set. Given a domain name d, we extract seventeen statistical features that describe the properties of the set RHDNs of domain names related to d. We divide these seventeen features into two groups:

- *String features.* This group consists of twelve features. We measure the number of distinct domain names in RHDNs, and the average and standard deviation of their length; the mean, median, and standard deviation of the occurrence frequency of each single character in the domain name strings in RHDNs; the mean, median and standard deviation of the distribution of 2-grams (i.e., pairs of characters); the mean, median and standard deviation of the distribution of

3-grams.

- *TLD features.* This group consists of five features. For each domain d_i in the RHDNs set, we extract its top-level domain $TLD(d_i)$ and we count the number of distinct TLD strings that we obtain; we measure the ratio between the number of domains d_i whose $TLD(d_i)$=".com" and the total number of TLD different from ".com"; also, we measure the mean, median, and standard deviation of the occurrence frequency of the TLD strings.

It is worth noting that whenever we measure the mean, median and standard deviation of a certain property, we do so in order to summarize the shape of its distribution. For example, by measuring the mean, median, and standard deviation of the occurrence frequency of each character in a set of domain name strings, we summarize what the distribution of the character frequency looks like.

3.2.2.3 Evidence-based Features

We use the evidence-based features to determine to what extent a given domain d is associated with other known malicious domain names or IP addresses. As mentioned above, Notos collects a *knowledge base* of known suspicious, malicious, and legitimate domain names and IPs from public sources. For example, we collect malware-related domain names by executing large numbers of malware samples in a controlled environment. Also, we check IP addresses against a number of public IP blacklists. We elaborate on how we build Notos' knowledge base in Section 3.3. Given a domain name d, we measure six statistical features using the information in the knowledge base. We divide these features into two groups:

- *Honeypot features.* We measure three features, namely the number of distinct malware samples that, when executed, try to contact d or any IP address in $A(d)$; the number of malware samples that contact any IP address

in $BGP(A(d))$; and the number of samples that contact any IP address in $AS(A(d))$.

- *Blacklist features.* We measure three features, namely the number of IP addresses in $A(d)$ that are listed in public IP blacklists; the number of IPs in $BGP(A(d))$ that are listed in IP blacklists; and the number of IPs in $AS(A(d))$ that are listed in IP blacklists.

Notos uses the blacklist features from the evidence vector so it can identify the reuse of known malicious network resources like IPs, BGP prefixes or even ASs. Domain names are significantly cheaper than IPv4 addresses; so malicious users tend to reuse address space with new domain names. We should note that the evidence-based features represent only part of the information we used to compute the reputation scores. The fact that a domain name was queried by malware does not automatically mean that the domain will receive a low reputation score.

3.2.3 Reputation Engine

Notos' reputation engine is responsible for deciding whether a domain name d has characteristics that are similar to either legitimate or malicious domain names. In order to achieve this goal, we first need to *train* the engine to recognize whether d belongs (or is "close") to a known *class of domains*. This training can be repeated periodically, in an off-line fashion, using historical information collected in Notos' *knowledge base* (see Section 3.3). Once the engine has been trained, it can be used in on-line mode to assign a reputation score to each new domain name d.

In this section, we first explain how the reputation engine is trained, and then we explain how a trained engine is used to assign reputation scores.

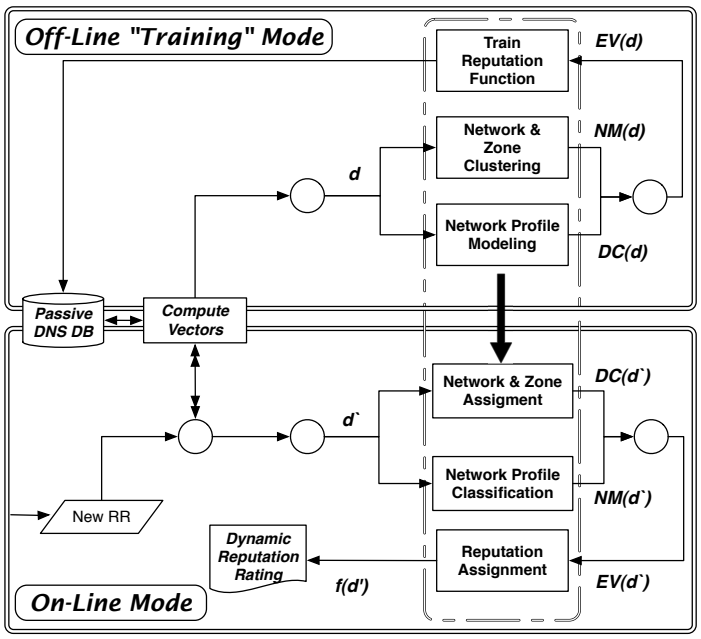


Figure 5: Notos' modes. The `Off-Line Mode` is necessary for training Notos. In the `On-Line Mode` Notos can dynamically assign reputation score to new RRs observed by our system.

3.2.3.1 Off-Line Training Mode

During off-line training (Figure 5), the reputation engine builds three different modules. We briefly introduce each module and then elaborate on the details.

- *Network Profiles Model*: a model of how well known networks behave. For example, we model the network characteristics of popular content delivery networks (e.g., Akamai, Amazon CloudFront), and large *popular* websites (e.g., google.com, yahoo.com). During the on-line mode, we compare each new domain name d to these models of well-known network profiles, and use this information to compute the final reputation score, as explained below.

- *Domain Name Clusters*: we group domain names into clusters sharing similar characteristics. We create these clusters of domains to identify groups of domains that contain mostly malicious domains, and groups that contain mostly

legitimate domains. In the on-line mode, given a new domain d, if d (more precisely, d's projection into a statistical feature space) falls within (or close to) a cluster of domains containing mostly malicious domains, for example, this gives us a hint that d should be assigned a low reputation score.

- *Reputation Function*: for each domain name $d_i, i = 1..n$, in Notos' knowledge base, we *test* it against the trained network profiles model and domain name clusters. Let $NM(d_i)$ and $DC(d_i)$ be the output of the Network Profiles (NP) module and the Domain Clusters (DC) module, respectively. The reputation function takes in input $NM(d_i)$, $DC(d_i)$, and information about whether d_i and its resolved IPs $A(d_i)$ are known to be legitimate, suspicious, or malicious (i.e., if they appeared in a domain name or IP blacklist), and builds a model that can assign a reputation score between zero and one to d. A reputation score close to zero signifies that d is a malicious domain name while a score close to one signifies that d is benign.

We now describe each module in detail.

3.2.3.2 Modeling Network Profiles

During the off-line training mode, the reputation engine builds a model of well-known network behaviors. An overview of the network profile modeling module can be seen in Figure 4(a). In practice we select five sets of domain names that share similar characteristics, and *learn* their network profiles. For example, we identify a set of domain names related to very popular websites (e.g., google.com, yahoo.com, amazon.com) and for each of the related domain names we extract their network features, as explained in Section 3.2.2.1. We then use the extracted feature vectors to train a statistical classifier that will be able to recognize whether a new domain name d has network characteristics similar to the popular websites we modeled.

In our current implementation of Notos we model the following classes of domain names:

- **Popular Domains**. This class consists of a large set of domain names under the following DNS zones: google.com, yahoo.com, amazon.com, ebay.com, msn.com, live.com, myspace.com, and facebook.com.

- **Common Domains**. This class of domains includes domain names under the top one hundred zones, according to `alexa.com`. We exclude from this group all the domain names already included in the *Popular Domains* class (which we model separately).

- **Akamai Domains**. Akamai is a large content delivery network (CDN), and the domain names related to this CDN have very peculiar network characteristics. To model the network profile of Akamai's domain names, we collect a set of domains under the following zones: akafms.net, akamai.net, akamaiedge.net, akamai.com, akadns.net, and akamai.com.

- **CDN Domains**. In this class we include domain names related to CDNs other than Akamai. For example, we collect domain names under the following zones: panthercdn.com, llnwd.net, cloudfront.net, nyud.net, nyucd.net and redcondor.net. We chose not to aggregate these CDN domains and Akamai's domains in one class, since we observed that Akamai's domains have a very unique network profile, as we discuss in Section 3.3. Therefore, learning two separate models for the classes of *Akamai Domains* and *CDN Domains* allows us to achieve better classification accuracy during the on-line mode, compared to learning only one model for both classes (see Section 3.2.3.5).

- **Dynamic DNS Domains**. This class includes a large set of domain names registered under two of the largest dynamic DNS providers, namely No-IP

(`no-ip.com`) and DynDNS (`dyndns.com`).

For each class of domains, we train a statistical classifier to distinguish between one of the classes and all the others. Therefore, we train five different classifiers. For example, we train a classifier that can distinguish between the class of *Popular Domains* and all other classes of domains. That is, given a new domain name d, this classifier is able to recognize whether d's network profile looks like the profile of a well-known popular domain or not. Following the same logic, we can recognize network profiles for the other classes of domains.

3.2.3.3 Building Domain Name Clusters

In this phase, the reputation engine takes the domain names collected in our pDNS database during a *training period*, and builds clusters of domains that share similar network and zone based features. The overview of this module can be seen in Figure 4(b). We perform clustering in two steps. In the first step we only use the network-based features to create coarse-grained clusters. Then, in the second step, we split each coarse-grained cluster into finer clusters using only the zone-based features, as shown in Figure 6.

Network-based Clustering The objective of network-based clustering is to group domains that share similar levels of *agility*. This creates separate clusters of domains with "stable" network characteristics and "non-stable" networks (like CDNs and malicious flux networks).

Zone-based Clustering After clustering the domain names according to their network-based features, we further split the network-based clusters of domain names into finer groups. In this step, we group domain names that are in the same network-based cluster and also share similar zone-based features. To better understand how

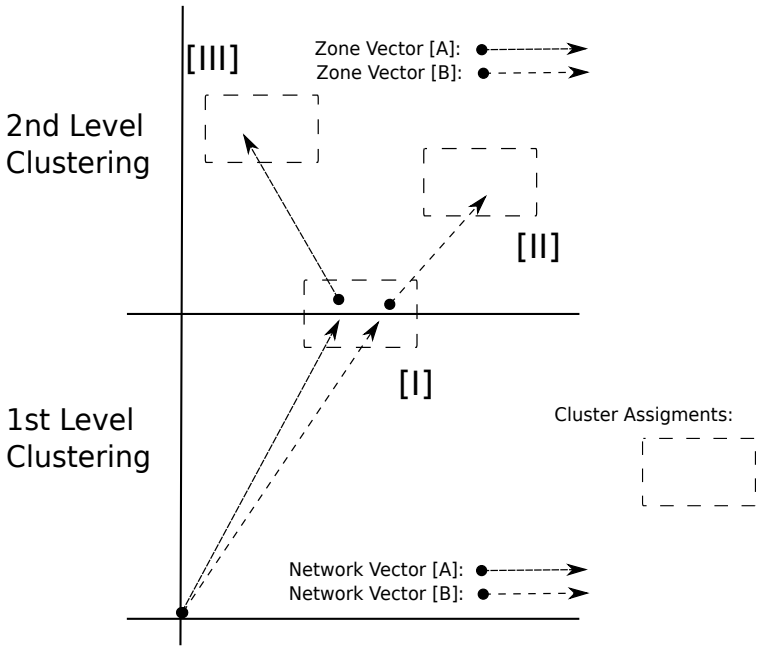


Figure 6: Network & zone based clustering process in Notos, in the case of a Akamai [A] and a malicious [B] domain name.

the zone-based clustering works, consider the following examples of zone-based clusters:

Cluster 1:

..., **72.247.176.81 e55.g.akamaiedge.net,** 72.247.176.94 e68.g.akamaiedge.net, 72.247.176.146 e120.g.akamaiedge.net, 72.247.176.65 e39.na.akamaiedge.net, 72.247.176.242 e216.g.akamaiedge.net, 72.247.176.33 e7.g.akamaiedge.net, 72.247.176.156 e130.g.akamaiedge.net, 72.247.176.208 e182.g.akamaiedge.net, 72.247.176.198 e172.g.akamaiedge.net, 72.247.176.217 e191.g.akamaiedge.net, 72.247.176.200 e174.g.akamaiedge.net, 72.247.176.99 e73.g.akamaiedge.net, 72.247.176.103 e77.g.akamaiedge.net, 72.247.176.59 e33.c.akamaiedge.net, 72.247.176.68 e42.gb.akamaiedge.net, 72.247.176.237 e211.g.akamaiedge.net, 72.247.176.71 e45.g.akamaiedge.net, 72.247.176.239 e213.na.akamaiedge.net, 72.247.176.120 e94.g.akamaiedge.net, ...

Cluster 2:

..., **90.156.145.198 spzr.in,** 90.156.145.198 vwui.in, 90.156.145.198 x9e.ru, 90.156.145.50 v2802.vps.masterhost.ru, 90.156.145.167 www.inshaker.ru, 90.156.145.198 x7l.ru, 90.156.145.198 c3q.at, 90.156.145.198 ltkq.in, 90.156.145.198 x7d.ru, 90.156.145.198 zdlz.in, 90.156.145.159 www.designcollector.ru, 90.156.145.198 x7o.ru, 90.156.145.198 q5c.ru, 90.156.145.159 designtwitters.com, 90.156.145.198 u5d.ru, 90.156.145.198 x9d.ru, 90.156.145.198 xb8.ru, 90.156.145.198 xg8.ru, 90.156.145.198 x8m.ru, 90.156.145.198 shopfilmworld.cn, 90.156.145.198 bigappletopworld.cn, 90.156.145.198 uppd.in, ...

Each element of the cluster is a *domain name - IP address* pair. These two groups

of domains belonged to the same network cluster, but were separated into two different clusters by the zone-based clustering phase. *Cluster 1* contains domain names belonging to Akamai's CDN, while the domains in *Cluster 2* are all related to malicious websites that distribute malicious software. The two clusters of domains share similar network characteristics, but have significantly different zone-based features. For example, consider domain names d_1="e55.g.akamaiedge.net" from the first cluster, and d_2="spzr.in" from the second cluster. The reason why d_1 and d_2 were clustered in the same network-based cluster is because the set of RHIPs (see Section 3.2.1) for d_1 and d_2 have similar characteristics. In particular, the *network agility* properties of d_2 make it look like if it was part of a large CDN. However, when we consider the set of RHDNs for d_1 and d_2, we can notice that the zone-based features of d_1 are much more "stable" than the zone-based features of d_2. In other words, while the RHDNs of d_1 share strong domain name similarities (e.g., they all share the substring "akamai") and have low variance of the *string features* (see Section 3.2.2.2), the strong *zone agility* properties of d_2 affect the zone-based features measured on d_2's RHDNs and make d_2 look very different from d_1.

One of the main advantages of Notos is the reliable assignment of low reputation scores to domain names participating in "agile" malicious campaigns. Less agile malicious campaigns, e.g., Fake anti-virus campaigns may use domain names structured to resemble CDN related domains. Such strategies would not be beneficial for the Fake anti-virus campaign, since domains like `virus-scan1.com`, `virus-scan2.com`, etc., can be trivially blocked by using simple regular expressions [62]. In other words, the attackers need to introduce more "agility" at both the network and domain name level in order to avoid simple domain name blacklisting. Notos would only require a few labeled domain names belonging to the malicious campaign for training purposes, and the reputation engine would then generalize to assign a low reputation score to the remaining (previously unknown) domain names that belong to the same malicious

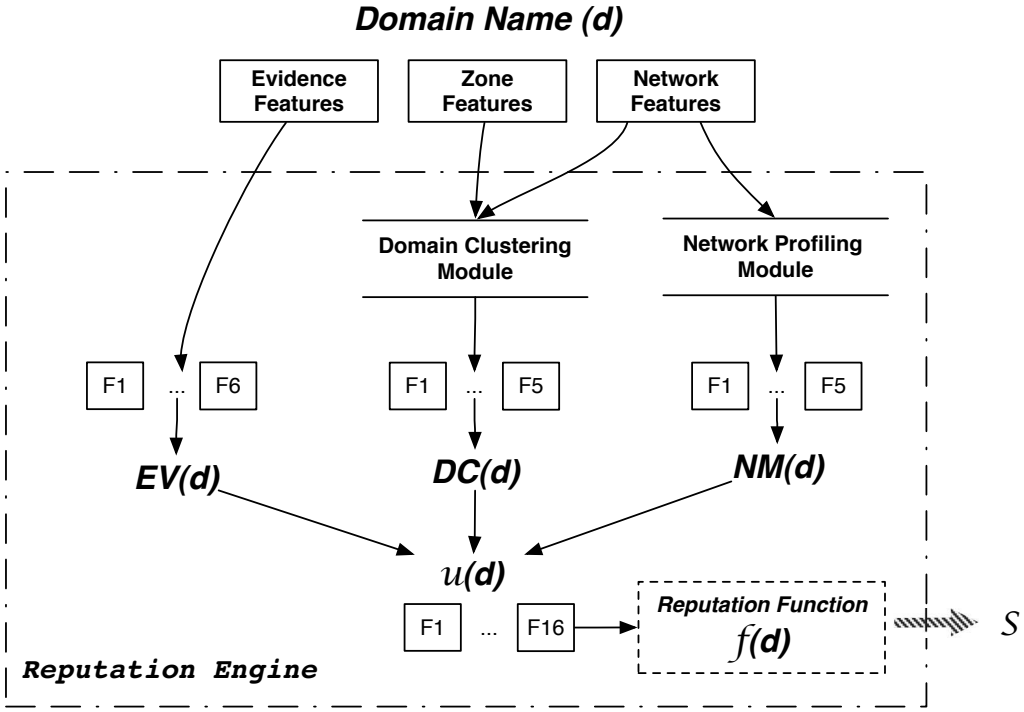


Figure 7: The output from the network profiling module, the domain clustering module and the evidence vector will assist the reputation function to assign the reputation score to the domain d.

campaign.

3.2.3.4 Building the Reputation Function

Once we build a model of well-known network profiles (see Section 3.2.3.2) and the domain clusters (see Section 3.2.3.3), we can build the reputation function. The reputation function will assign a reputation score in the interval $[0, 1]$ to domain names, with 0 meaning low reputation (i.e., likely malicious) and 1 meaning high reputation (i.e., likely legitimate). We implement our reputation function as a statistical classifier. In order to train the reputation function, we consider all the domain names $d_i, i = 1, .., n$ in Notos' *knowledge base*, and we feed each domain d_i to the *network profiles* module and to the *domain clusters* module to compute two output vectors $NM(d_i)$ and $DC(d_i)$, respectively. We explain the details of how $NM(d_i)$ and $DC(d_i)$

are computed later in Section 3.2.3.5. For now it sufficient to consider $NM(d_i)$ and $DC(d_i)$ as two feature vectors. For each d_i we also compute an *evidence features* vector $EV(d_i)$, as described in Section 3.2.2.3. Let $v(d_i)$ be a feature vector that combines the $NM(d_i)$, $DC(d_i)$, and $EV(d_i)$ feature vectors. We train the reputation function using the labeled dataset $L = \{(v(d_i), y_i)\}_{i=1..n}$, where $y_i = 0$ if d_i is a known malicious domain name, otherwise $y_i = 1$.

3.2.3.5 On-Line Mode

After training is complete; the reputation engine can be used in on-line mode (Figure 5) to assign a reputation score to new domain names. For example, given an input domain name d, the reputation engine computes a score $S \in [0, 1]$. Values of S close to zero mean that d appears to be related to malicious activities and therefore has a low reputation. On the other hand, values of S close to one signify that d appears to be associated with benign Internet services, and therefore has a high reputation. The reputation score is computed as follows. First, d is fed into the *network profiles* module, which consists of five statistical classifiers, as discussed in Section 3.2.3.2. The output of the *network profiles* module is a vector $NM(d) = \{c_1, c_2, ..., c_5\}$, where c_1 is the output of the first classifier, and can be viewed as the probability that d belongs to the class of *Popular Domains*, c_2 is the probability that d belongs to the class of *Common Domains*, etc. At the same time, d is fed into the *domain clusters* module, which computes a vector $DC(d) = \{l_1, l_2, ..., l_5\}$. The elements l_i of this vector are computed as follows. Given d, we first extract its network-based features and identify the closest network-based cluster to d, among the network-based clusters computed by the *domain clusters* module during the off-line mode (see Section 3.2.3.3). Then, we extract the zone-based statistical features and identify the zone-based cluster closest to d. Let this closest domain cluster be C_d. At this point, we consider all the zone-based feature vectors $v_j \in C_d$, and we select the subset of vectors $V_d \subseteq C_d$ for

which the two following conditions are verified: i) $dist(z_d, v_j) < R$, where z_d is the zone-based feature vector for d, and R is a predefined *radius*; ii) $v_j \in KNN(z_d)$, where $KNN(z_d)$ is the set of k nearest-neighbors of z_d.

The feature vectors in V_d are related to domain names extracted from Notos' *knowledge base*. Therefore, we can assign a label to each vector $v_i \in V_d$, according to the nature of the domain name d from which v_i was computed. The domains in Notos' *knowledge base* belong to different classes. In particular, we distinguish between eight different classes of domains, namely *Popular Domains*, *Common Domains*, *Akamai*, *CDN*, and *Dynamic DNS*, which have the same meaning as explained in Section 3.2.3.2, and *Spam Domains*, *Flux Domains*, and *Malware Domains*.

In order to compute the output vector $DC(d)$, we compute the following five statistical features: the *majority class* label L (e.g., L may be equal to *Malware Domain*), i.e., the label that appears the most among the vectors $v_i \in V_d$; the standard deviation of label frequencies, i.e., given the occurrence frequency of each label among the vectors $v_i \in V_d$ we compute their standard deviation; given the subset $V_d^{(L)} \subseteq V_d$ of vectors in V_d that are associated with label L, we compute the *mean*, *median* and *standard deviation* of the distribution of distances between z_d and the vectors $v_j \in V_d^{(L)}$.

3.2.3.6 Assigning Reputation Scores

Given a domain d, once we compute the vectors $NM(d)$ and $DC(d_i)$ as explained above, we also compute the evidence vector $EV(d)$ as explained in Section 3.2.2.3. At this point, we concatenate these three feature vectors into a sixteen dimensional feature vector $v(d)$, and we feed $v(d)$ in input to our *trained* reputation function (see Section 3.2.3.4). The reputation function computes a score $S = 1 - f(d)$, where $f(d)$ can be interpreted as the probability that d is a malicious domain name. S varies in the $[0, 1]$ interval, and the smaller the value of S, the lower d's reputation.

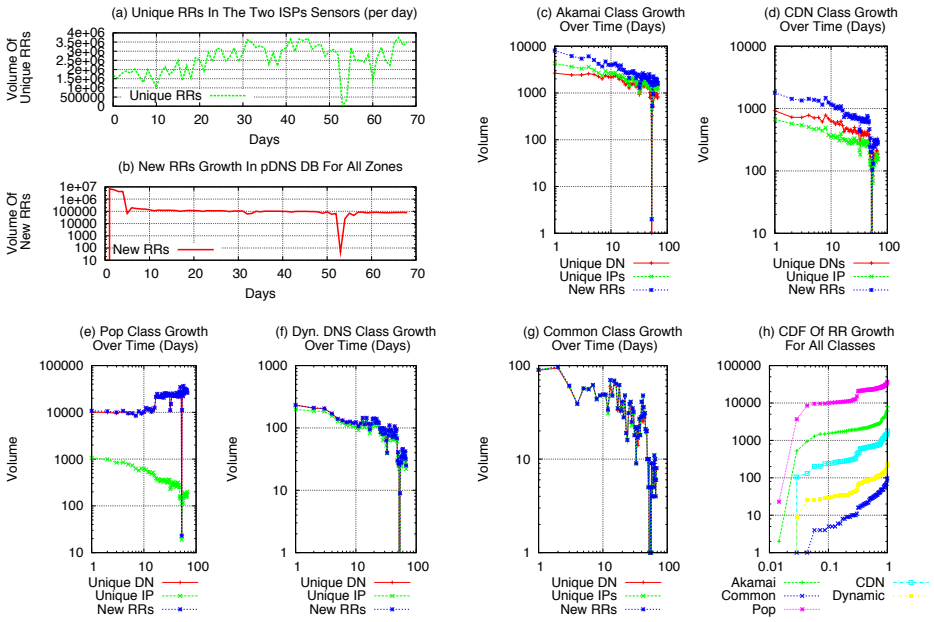


Figure 8: Various RRs growth trends observed in the pDNS DB over a period of 68 days

3.3 Data Collection

This section summarizes observations from passive DNS measurements, and how professional, legitimate DNS services are distinguished from malicious services. These observations provided the ground truth for our dynamic domain name reputation system. We also provide an intuitive example to illustrate these properties, using a few major Internet zones like Akamai and Google.

3.3.1 Data Collection

The basic building block for our dynamic reputation rating system is the historical or "passive" information from successful `A-type` DNS resolutions. We use the DNS traffic from two ISP-based sensors, one located on the US east coast (Atlanta) and one located on the US west coast (San Jose). Additionally we use the aggregated DNS traffic from the different networks covered by SIE [17]. In total, our database collected 27,377,461 unique resolutions from all these sources over a period of 68 days,

from 19^{th} of July 2009 to 24^{th} September 2009.

Simple measurements performed on this large data set demonstrate a few important properties leveraged by our selected features. After just a few days the rate of new, unique pDNS entries leveled off. The graph in Figure 8(b) shows only about 100,000 to 150,000 new domains/day (with a brief outage issue on the 53^{rd} day), despite very large numbers of RRs arriving each day (shown in Figure 8(a)). This suggests that most RRs are duplicates, and approximately after the first few days, 94.7% – on average – from the unique RRs observed in daily base at the sensor level are already recorded by the passive DNS database. Therefore, even a relatively small pDNS database may be used to deploy Notos. In Section 3.5, we measure the sensitivity of our system to traffic collected from smaller networks.

The remaining plots in Figure 8 show the daily growth of our passive DNS database, from the point of view of five different zone classes. Figure 8(c) and (d) show the growth rate associated with CDN networks (Akamai, and all other CDNs). The number of unique IP addresses stays nearly constant with the number of unique domains (meaning that each new RR is a new IP address and a new child domain of the CDN). In a few weeks, most of the IP addresses became known—suggesting that one can fully map CDNs in a modest training set. This is because CDNs, although large, always have a fixed number of IP addresses used for hosting their high-availability services. Intuitively, we believe this would not be the case with malicious CDNs (e.g., flux networks), which use randomly spreading infections to continually recruit new IP addresses.

The ratio of new IP addresses to domains diverges in Figure 8(e), a plot of the rate of newly discovered RRs for popular websites (e.g., Google, Facebook). Facebook notably uses unique child domains for their Web-based chat client, and other top Internet sites use similar strategies (encoding information in the domain, instead of the URI), which explains the growth in domains shown in Figure 8(e). These

popular sites use a very small number of IP addresses, however, and after a few weeks of training our pDNS database identified all of them. Since these popular domains make up a large portion of traffic in any trace, our intuition is that simple whitelisting would significantly reduce the workload of a classifier.

Figure 8(f) shows the rate of pDNS growth for zones in Dynamic DNS providers. These services, sometimes used by botmasters, demonstrate a nearly matched ratio of new IP addresses to new domains. The data excludes non-routable answers (e.g., dynamic DNS domains pointing to 127.0.0.1), since this contains no unique network information. Intuitively, one can think of dynamic DNS as a nearly complete bijection of domains to IP addresses. Figure 8(g) shows the growth of RRs for the `alexa.com` top 100 domains. Unlike dynamic DNS domains, these point to a small set of unique addresses, and most can be identified in a few weeks' worth of training.

A comparison of all the zone classes appears in Figure 8(h), which shows the cumulative distribution of the unique RRs detailed in Figure 8(c) through 8(g). The different rates of change illustrate how each zone class has a distinct pattern of RR use: some have a small IP address space and highly variable domain names; some pair nearly every new domain with a new IP address. Learning approximately 90% of all the unique RRs in each zone class, however, only requires (at most) tens of thousands of distinct RRs. The intuition from this plot is that, despite the very large data set we used in our study, Notos could potentially work with data observed from much smaller networks.

3.3.2 Building The Ground Truth

To establish ground truth, we use two different labeling processes. First, we assigned labels to RRs at the time of their discovery. This provided an initial static label for many domains. Blacklists, of course, are never complete and always dynamic. So our second labeling process took place during evaluation, and monitored several

well-known domain blacklists and whitelists.

The data we used for labeling came from several sources. Our primary source of blacklisting came from services such as *malwaredomainlist* [1] and *malwaredomains* [2]. In order to label IP addresses in our pDNS database we also used the Sender Policy Block (SBL) list from Spamhaus [67]. Such IP addresses are either known to send spam or distribute malware. We also collected domain name and IP address blacklisting information from the Zeus tracker [108]. All this blacklisting information was gathered before the first day of August 2009 (during all the 15 days in which we collected passive DNS data). Since blacklists traditionally lag behind the active threat, we continued to collect all new data until the end of our experiments.

Our limited whitelisting was derived from the top 500-`alexa.com` domain names, as of the 1^{st} of August 2009. We reasoned that, although some malicious domains become popular, they do not stay popular (because of remediation), and never break into the top tier of domain rankings. Likewise, we used a list of the 18 most common 2LDs from various CDNs, which composed the main corpus of our CDN labeled RRs. Finally a list of 464 dynamic DNS second level domains allowed us to identify and label domain name and IP addresses coming from zones under dynamic DNS providers. We label our evaluation (or testing) data-set by aggregating updated blacklist information for new malicious domain names and IP addresses from the same lists.

To compute the honeypot features (presented in Section 3.2.2.3) we need a malware analysis infrastructure that can process as many "new" malware samples as possible. Our honeypot infrastructure is similar to "Ether" [22] and is capable of processing malware samples in a queue. Every malware sample was analyzed in a controlled environment for a time period of five minutes. This process was repeated during the last 15 days of July 2009. After 15 days of executions we obtained a set

[1] malwaredomainlist.com
[2] malwaredomains.com

Table 1: Network-based features (NF) description

NF_1	The freq. count of unique ASs in $RHIPS_d$
NF_2	The freq. count of unique CIDRs in $RHIPS_d$
NF_3	The freq. count of unique CCs in $RHIPS_d$
NF_4	The freq. count of unique registrars in $RHIPS_d$
NF_5	The freq. count of unique registration dates in $RHIPS_d$
NF_6	The freq. count of unique owners in $RHIPS_d$
NF_7	The freq. count of unique IPs in the $RHIPS_{2LD(d)}$
NF_8	The freq. count of unique IPs in the $RHIPS_{3LD(d)}$
NF_9	The freq. count of unique CIDRs in the $RHIPS_{2LD(d)}$
NF_{10}	The freq. count of unique CIDRs in the $RHIPS_{3LD(d)}$
NF_{11}	The freq. count of unique ASs in the $RHIPS_{2LD(d)}$
NF_{12}	The freq. count of unique ASs in the $RHIPS_{3LD(d)}$
NF_{13}	The freq. count of unique owners in the $RHIPS_{2LD(d)}$
NF_{14}	The freq. count of unique owners in the $RHIPS_{3LD(d)}$
NF_{15}	The freq. count of unique CCs in the $RHIPS_{2LD(d)}$
NF_{16}	The freq. count of unique CCs in the $RHIPS_{3LD(d)}$
NF_{17}	The freq. count of unique registration dates in the $RHIPS_{2LD(d)}$
NF_{18}	The freq. count of unique registration dates in the $RHIPS_{3LD(d)}$

of successful DNS resolutions (domain names and IP addresses) that each malware looked up. We chose to execute malware and collect DNS evidence through the same period of time in which we aggregate the passive DNS database. Our virtual machines are equipped with five popular commercial anti-virus engines. If one of the engines identifies an executable as malicious, we capture all domain names and the corresponding IP addresses mappings that the malware used during execution. After excluding all domain names that belong to the top 500 most popular `alexa.com` zones, we assemble the main corpus of our "honeypot data". We automated the crawling and collection of black list information and honeypot execution.

The reader should note that we chose to label our data in as transparent a way as possible. We used public blacklisting information to label our training dataset before we build our models and train the reputation function. Then we assigned the reputation scores and validated the results again using the same publicly available blacklist sources. It is safe to assume that private IP and DNS blacklist will contain significantly more complete information with lower FP rates than the public blacklists. By using such a type of private blacklist the accuracy of Notos' reputation function should improve significantly.

3.4 Statistical Feature Analysis

Next we provide an in-depth analysis of all statistical features used in Notos. The first step towards evaluating the importance of each feature used by Notos is to compute the mutual information matrix [25] for each different statistical vector (Section 3.2.1, Figure 3 Network, Zone and Reputation Vectors). Each matrix measures the *mutual dependence* of pairs of statistical features in the vector.

The 18-dimensional network vector is used during the clustering and classification process. Detailed description of the statistical features we used can be found in Table 1. As a quick reminder, with this vector we try to capture a number of different network properties for each RR and its effective 2LD and 3LD (if available). We need to capture these characteristics by the 18 statistical features selected in the network-based vector. The Table 2 and the upper left heat-plot from Figure 9 we can see that almost all selected features in our network vector are strongly independent. One of the most interesting observation we can make from the mutual information matrix is that the features computed based on IP-based properties (features 1 to 6) are strongly independent from the features that have been computed based on the CIDR and AS information (features 7-18). Also there is some level of dependence between features computed on CIDR and AS information. This implies that we could reduce the number of features computed based on CIDR and AS information without significantly affecting the discriminatory ability of the network-based features.

With the 17-dimensional zone based vector we try to capture characteristics about the zones that have been historically hosted in an IP address. Several characteristics around the observed zones will help us create a profile about the address space utilization with respect to the zones hosted in that space (i.e., the number of distinct 2LDs and 3LDs historically can be associated with an IP). Detailed description of the statistical features we used to assemble the zone-based vectors can be found in Table 4.

Table 2: Mutual information matrix for network-based features (NF)

	IP Based Features						CIDR and AS Based Features										
	NF_1	NF_2	NF_3	NF_4	NF_5	NF_6	NF_7	NF_8	NF_9	NF_{10}	NF_{11}	NF_{12}	NF_{13}	NF_{14}	NF_{15}	NF_{16}	NF_{17}
NF_2	0.705																
NF_3	0.701	0.511															
NF_4	0.635	0.469	0.892														
NF_5	0.759	0.866	0.577	0.5220													
NF_6	0.928	0.759	0.659	0.602	0.738												
NF_7	0.139	0.210	0.108	0.118	0.17079	0.165											
NF_8	0.102	0.186	0.081	0.107	0.15210	0.120	0.713										
NF_9	0.205	0.272	0.161	0.164	0.23278	0.232	0.858	0.603									
NF_{10}	0.144	0.248	0.110	0.139	0.20329	0.165	0.597	0.857	0.559								
NF_{11}	0.249	0.278	0.196	0.195	0.24925	0.275	0.792	0.550	0.915	0.510							
NF_{12}	0.164	0.234	0.122	0.154	0.20576	0.182	0.561	0.835	0.510	0.940	0.497						
NF_{13}	0.238	0.283	0.187	0.187	0.24792	0.266	0.811	0.571	0.938	0.533	0.977	0.515					
NF_{14}	0.161	0.235	0.120	0.152	0.20299	0.184	0.567	0.836	0.521	0.950	0.500	0.988	0.525				
NF_{15}	0.321	0.390	0.298	0.280	0.36043	0.359	0.465	0.343	0.542	0.396	0.589	0.385	0.583	0.387			
NF_{16}	0.130	0.201	0.138	0.171	0.18804	0.140	0.512	0.794	0.440	0.853	0.412	0.893	0.432	0.883	0.398		
NF_{17}	0.232	0.294	0.184	0.182	0.27814	0.260	0.829	0.588	0.936	0.549	0.913	0.526	0.917	0.532	0.619	0.465	
NF_{18}	0.141	0.220	0.115	0.145	0.21477	0.148	0.575	0.862	0.515	0.943	0.481	0.931	0.498	0.928	0.382	0.890	0.545

Table 3: Mutual information matrix for domain name or zone based features (ZF)

	Domain Name (or Zone) Based Features															
	ZF_1	ZF_2	ZF_3	ZF_4	ZF_5	ZF_6	ZF_7	ZF_8	ZF_9	ZF_{10}	ZF_{11}	ZF_{12}	ZF_{13}	ZF_{14}	ZF_{15}	ZF_{16}
ZF_2	0.00019															
ZF_3	0.11667	0.2078														
ZF_4	0.11487	0.3400	0.1376													
ZF_5	0.11643	0.2944	0.0311	0.5319												
ZF_6	0.17154	0.2941	0.0079	0.5352	0.9411											
ZF_7	0.14662	0.3586	0.0071	0.5360	0.8780	0.9662										
ZF_8	0.06757	0.3918	0.0312	0.4949	0.8773	0.8918	0.9029									
ZF_9	0.12263	0.3147	0.0043	0.4661	0.8746	0.9353	0.9469	0.9178								
ZF_{10}	0.12033	0.3165	0.0031	0.4852	0.8766	0.9493	0.964	0.9028	0.9884							
ZF_{11}	0.00080	0.0080	0.0528	0.0191	0.0408	0.0225	0.0093	0.0187	0.0102	0.0118						
ZF_{12}	0.07385	0.3865	0.0201	0.5181	0.8438	0.8868	0.9145	0.9495	0.9329	0.9225	0.00471					
ZF_{13}	0.11770	0.3229	0.0027	0.4879	0.8563	0.9171	0.9390	0.8903	0.9814	0.9756	0.00410	0.9363				
ZF_{14}	0.11436	0.3198	0.0022	0.4957	0.8646	0.9351	0.9551	0.8886	0.9771	0.9874	0.00706	0.9294	0.9909			
ZF_{15}	0.14553	0.2142	0.0349	0.4062	0.6988	0.7881	0.7893	0.6983	0.7952	0.8153	0.00466	0.7319	0.8113	0.8242		
ZF_{16}	0.13359	0.2860	0.0019	0.5585	0.8377	0.9093	0.9118	0.8451	0.9103	0.9285	0.01255	0.8694	0.9206	0.9357	0.9189	
ZF_{17}	0.08655	0.2133	0.9022	0.1699	0.0507	0.0185	0.0171	0.0515	0.0127	0.0107	0.07125	0.0373	0.0101	0.009	0.0138	0.00047

Table 4: Zone-based features (ZF) description

ZF_1	The average domain length in $RHDNS$
ZF_2	The standard deviation of the domain length in $RHDNS$
ZF_3	The number of unique TLDs in the $RHDNS$
ZF_4	The number of unique characters in the 2LDs in the $RHDNS$
ZF_5	The median of the unique character freq. of the domains in the $RHDNS$
ZF_6	The average of the unique character freq. of the domains in the $RHDNS$
ZF_7	The standard deviation of the unique character freq. of the domains in the $RHDNS$
ZF_8	The median of the unique 2-grams freq. of the domains in the $RHDNS$
ZF_9	The average of the unique 2-grams freq. of the domains in the $RHDNS$
ZF_{10}	The standard deviation of the unique 2-grams freq. of the domains in the $RHDNS$
ZF_{11}	The ratio of non-COM TLDs over all TLDs in the $RHDNS$
ZF_{12}	The median of the unique 3-grams freq. of the domains in the $RHDNS$
ZF_{13}	The average of the unique 3-grams freq. of the domains in the $RHDNS$
ZF_{14}	The standard deviation of the unique 3-grams freq. of the domains in the $RHDNS$
ZF_{15}	The median of the unique TLDs freq. of the domains in the $RHDNS$
ZF_{16}	The median of the unique TLDs freq. of the domains in the $RHDNS$
ZF_{17}	The standard deviation of the unique TLDs freq. of the domains in the $RHDNS$

In the Table 3 and the second upper right heat-plot from Figure 9 we observe the mutual information between features in the zone based vectors. We can see that the set of features, 1 to 4, which have been computed based on observation from the stricture of domain names (in the RHDN) shows strong independence from the features, 4-15, that have been computed based on 2LD and 3LD observations. The exceptions in these observations are feature 11 (the ratio of non-COM TLDs over all TLDs) and feature 17 (the standard deviation of the unique TLDs frequency). From the heat-maps in Figure 9 it is clear that the network based features shows a larger degree of independence than the zone based features. This observation was one of reason that motivated our decision to make the first level clustering based on network based and not with the zone based vectors.

In Table 5 and the lower heat-plot in Figure 9 we can see the mutual information between the features in the reputation vector. The results shows that almost all features are independent. The only exception in this is the features (12-16) that hold the first order statistics from the distance between other (labeled) vectors in the same cluster.

Table 5: Mutual information matrix for reputation features (RF)

	Reputation Features														
	Supervised Vector Class Confidences				Evidence Vector Malware Traces			Evidence Vector Public BLs			Unsupervised Vector $DC(d)$				
	RF_1	RF_2	RF_3	RF_4	RF_5	RF_6	RF_7	RF_8	RF_9	RF_{10}	RF_{11}	RF_{12}	RF_{13}	RF_{14}	RF_{15}
RF_2	0.06480														
RF_3	0.08950	0.00766													
RF_4	0.00612	0.28121	0.01322												
RF_5	0.00936	0.29219	0.00751	0.08033											
RF_6	0.15241	0.27471	0.06360	0.05083	0.04059										
RF_7	0.00534	0.07048	0.00028	0.00825	0.00328	0.13893									
RF_8	0.00479	0.16089	0.00333	0.10806	0.07521	0.04387	0.02660								
RF_9	0.04174	0.00089	0.00361	0.00001	0.15116	0.01500	0.01179	0.02636							
RF_{10}	0.00018	0.00368	0.01271	0.00182	0.00112	0.04055	0.08464	0.00012	0.00074						
RF_{11}	0.00606	0.01225	0.01755	0.00028	0.00022	0.08473	0.05687	0.00191	0.00550	0.18245					
RF_{12}	0.00098	0.03492	0.03504	0.04418	0.08873	0.02141	0.01152	0.08234	0.01892	0.06345	0.29210				
RF_{13}	0.00455	0.02174	0.00727	0.00381	0.01910	0.00435	0.00031	0.03428	0.00688	0.00197	0.01325	0.01808			
RF_{14}	0.00535	0.02091	0.00722	0.00382	0.01442	0.00508	0.00005	0.03000	0.00504	0.00150	0.01100	0.01894	0.788		
RF_{15}	0.00473	0.02211	0.00768	0.00354	0.01925	0.00446	0.00032	0.03458	0.00675	0.00200	0.01338	0.01866	0.989	0.793	
RF_{16}	0.00055	0.00018	0.00008	0.00970	0.00389	0.00147	0.00880	0.00719	0.00270	0.00023	0.00596	0.00721	0.257	0.24	0.262

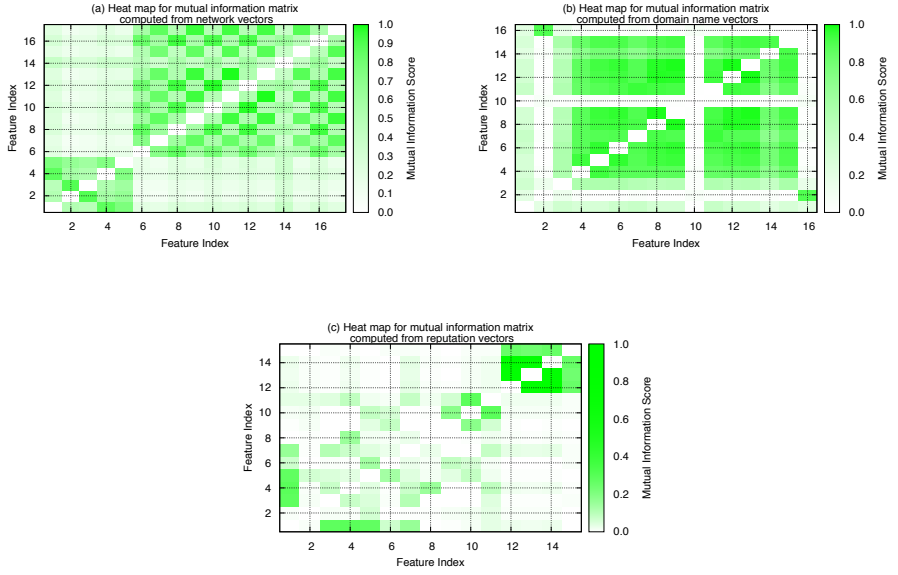


Figure 9: The mutual information from all different vectors used in Notos in heat-map projection.

3.5 Evaluation

In this section, we present the experimental results of our evaluation. We show that Notos can identify malicious domain names sooner than public blacklists, with a low false positive rate (FP%) of 0.38% and high true positive rate (TP%) of 96.8%. As a first step, we computed vectors based on the statistical features (described in Section 3.2.2) from 250,000 unique RRs. This volume corresponds to the average volume of new – previously unseen – RRs observed at two recursive DNS servers in a major ISP in one day, as noted in Section 3.3, Figure 8(b). These vectors were computed based on historic passive DNS information from the last two weeks of DNS traffic observed on the same two ISP recursive resolvers in Atlanta and San Jose.

3.5.1 Accuracy of Network Profile Modeling

The accuracy of the Meta-Classification system (Figure 10) in the network profile module is critical for the overall performance of Notos. This is because, in the on-line

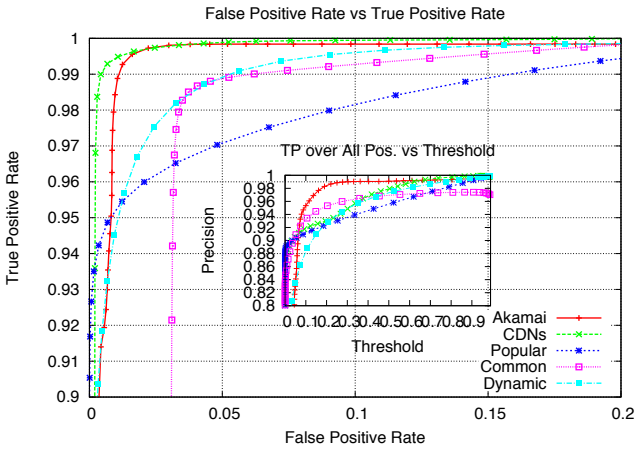


Figure 10: ROC curves for all network profile classes shows the Meta-Classifier's accuracy.

mode, Notos will receive unlabeled vectors which must be classified and correlated with what is already present in our knowledge base. For example, if the classifier receives a new RR and assigns to it the label Akamai with very high confidence, that implies the RR which produced this vector will be part of a network similar to Akamai. However, this does not necessarily mean that it is part of the actual Akamai CDN. We will see in the next section how we can draw conclusions based on the proximity between labeled and unlabeled RRs within the same zone-based clusters. Furthermore, we discuss the accuracy of the Meta-Classifier when modeling each different network profile class (profile classes are described in Section 3.2.3.2).

Our Meta-Classifier consists of five different classifiers, one for each different class of domains we model. We chose to use a Meta-Classification system instead of a traditional single classification approach because Meta-Classification systems typically perform better than a single statistical classifier [13, 44]. Throughout our experiments this proved to be also true. The ROC curve in Figure 10, shows that the Meta-Classifier can accurately classify RRs for all different network profile classes.

The training dataset for the Meta-Classifier is composed of sets of 2,000 vectors from each of the five network profile classes. The evaluation dataset is composed

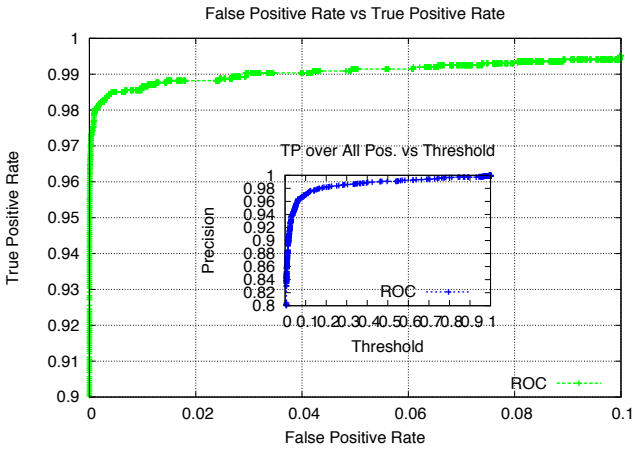


Figure 11: The ROC curve from the reputation function indicating the high accuracy of Notos.

of 10,000 vectors, 2,000 from each of the five network profile classes. The classification results for the domains in the Akamai, CDN, dynamic DNS and Popular classes showed that the supervised learning process in Notos is accurate, with the exception of a small number of false positives related to the Common class (3.8%). After manually analyzing these false positives, we concluded that some level of confusion between the vectors produced by Dynamic DNS domain names and the vectors produced by domain names in the Common class still remains. However, this minor misclassification between network profiles does not significantly affect the reputation function. This is because the zone profiles of the Common and Dynamic DNS domain names are significantly different. This difference in the zone profiles will drive the network-based and zone-based clustering steps to group the RRs from Dynamic DNS class and Common class in different zone-based clusters.

Despite the fact that the network profile modeling process provides accurate results, it doesn't mean this step can independently designate a domain as benign or malicious. The clustering steps will assist Notos to group vectors not only based their network profiles but also based on their zone properties. In the following section we show how the network and zone profile clustering modules can better associate similar

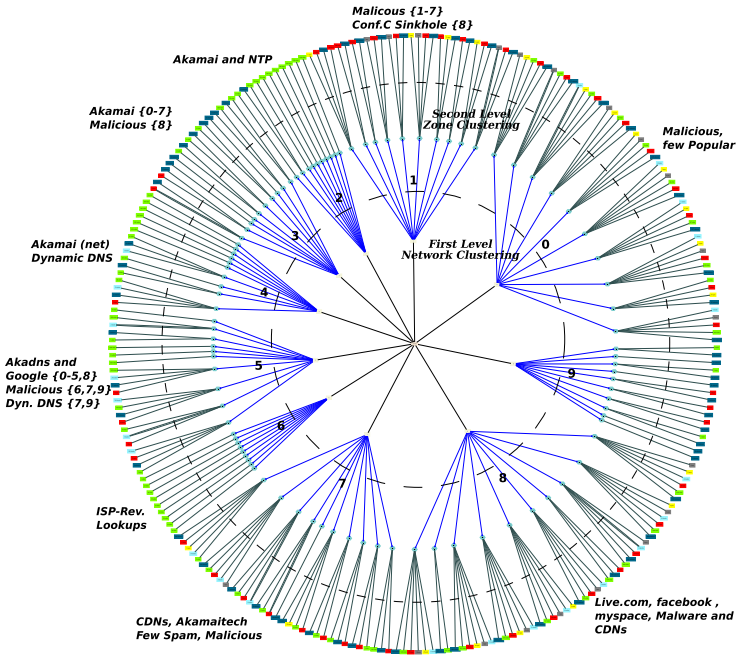


Figure 12: With the 2-step clustering step, Notos is able to cluster large trends of DNS behavior.

vectors, due to properties of their domain name structure.

3.5.2 Network and Zone-Based Clustering Results

In the domain name clustering process (Section 3.2.3.3, Figure 4(b)) we used X-Means clustering in series, once for the network-based clustering and again for the zone-based clustering. In both steps we set the minimum and maximum number of clusters to one and the total number of vectors in our dataset, respectively. We run these two steps using different numbers of zone and network vectors. Figure 13 shows that after the first 100,000 vectors are used, the number of network and zone clusters remains fairly stable. This means that by computing at least 100,000 network and zone vectors—using a 15-day old passive DNS database—we can obtain a stable population of zone and network based clusters for the monitored network. We should note that reaching this network and cluster equilibrium does not imply that we do not expect to see any new type of domain names in the ISP's DNS recursive. This just denotes that based

on the RRs present in our passive DNS database, and the daily traffic at the ISP's recursive, 100,000 vectors are enough to reflect the major network profile trends in the monitored networks.

Figure 13 indicates that a sample set of 100,000 vectors may represent the major trends in a DNS sensor. It is hard to safely estimate the exact minimum number of unique RRs that is sufficient to identify all major DNS trends. An answer to this should be based upon the type, size and utilization of the monitored network. Without data from smaller corporate networks it is difficult for us to make a safe assessment about the minimum number of RRs necessary for reliably training Notos.

The evaluation dataset we used consisted of 250,000 unique domain names and IP addresses. The cluster overview is shown in Figure 12 and in the following paragraphs we discuss some interesting observations that can be made from these network-based and zone-based cluster assignments. As an example, network clusters 0 and 1 are predominantly composed of zones participating in fraudulent activities like spam campaigns (yellow) and malware dropping or C&C zones (red). On the other hand, network clusters 2 to 5 contain Akamai, dynamic DNS, and popular zones like Google, all labeled as benign (green). We included the unlabeled vectors (blue) based on which we evaluated the accuracy of our reputation function. We have a sample of unlabeled vectors in almost all network and zone clusters. We will see how already labeled vectors will assist us to characterize the unlabeled vectors in close proximity.

Before we describe two sample cases of dynamic characterization within zone-based clusters, we need to discuss our radius R and k value selection (see Section 3.2.3.5). In Section 3.2.3.5, we discuss how we build domain name clusters. At that point we introduced the dynamic characterization process that gives Notos the ability to utilize already labeled vectors in order to characterize a newly obtained unlabeled vector by leveraging our prior knowledge. After looking into the distribution of Euclidean distances between unlabeled and labeled vectors within the same zone clusters, we

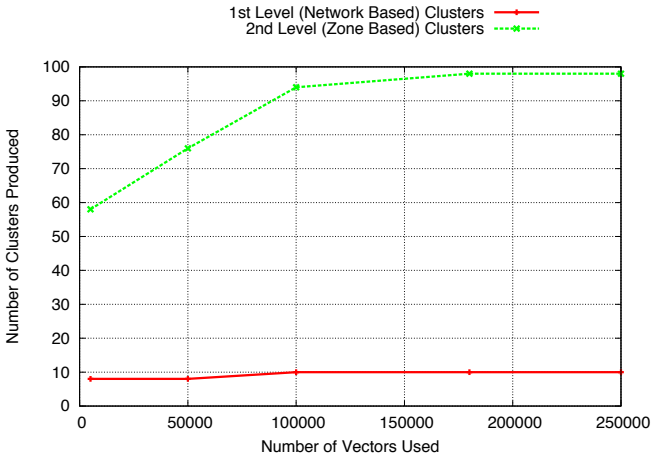


Figure 13: By using different number of network and zone vectors we observe that after the first 100,000, there is no significant variation in the absolute number of produced clusters during the 1^{st} and 2^{nd} level clustering steps.

concluded that in the majority of these cases the distances were between 0 and 1000. We tested different values of the radius R and the value of k for the K-nearest neighbors (KNN) algorithm. We observed that the experiments with radius values between 50 and 200 provided the most accurate reputation rating results, which we describe in the following sections. We also observed that if $k > 25$ the accuracy of the reputation function is not affected for all radius values between 50 and 200. Based on the results of these pilot experiments, we decided to set k equal to 50 and the radius distance equal to 100.

Figures 14 and 15 show the effect of this radius selection on two different types of clustering problems. In Figure 14, unknown RRs for `akamaitech.net` are clustered with a labeled vector `akamai.net`. As noted in Section 3.3, CDNs such as Akamai tended to have new domain names with each RR, but to also reuse their IP addresses. By training with only a small set of labeled `akamai.net` RRs, our classifier put the new, unknown RRs for `akamaitech.net` into the existing Akamai class. IP-specific features therefore brought the new RRs close to the existing labeled class. Figure 14 compresses all of the dimensions into a two-dimensional plot (for easier

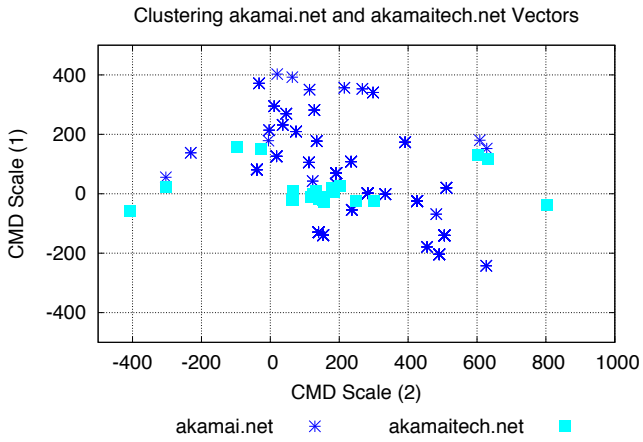


Figure 14: An example of characterizing the akamaitech.net unknown vectors as benign based on the already labeled vectors (akamai.net) present in the same cluster.

visual representation), but it is clear the unknown RRs were all within a distance of 100 to the labeled set.

This result validates our assumptions from Section 3.3, where just a few weeks' worth of labeled data was necessary for training. Thus, one does not have to exhaustively discover all whitelisted domains. Notos is resilient to changes in the zone classes we selected. Services like CDNs and major web sites can add new IP addresses or adjust domain formats, and these will be *automatically* associated with a known labeled class.

The ability of Notos to associate new RRs based on limited labeled inputs is demonstrated again in Figure 15. In this case, labeled Zeus domains (approximately 2,900 RRs from three different Zeus-related BLs) were used to classify new RRs. Figure 15 plots the distance between the labeled Zeus-related RRs and new (previously unknown) RRs that are also related Zeus botnets. As we can see from Section 3.3, most of the new (unlabeled) Zeus RRs lay very close, and often even overlap, to known Zeus RRs. This is a good result, because Zeus botnets are notoriously hard to track, given the botnet's extreme agility. Tracking systems such as `zeustracker.abuse.ch`

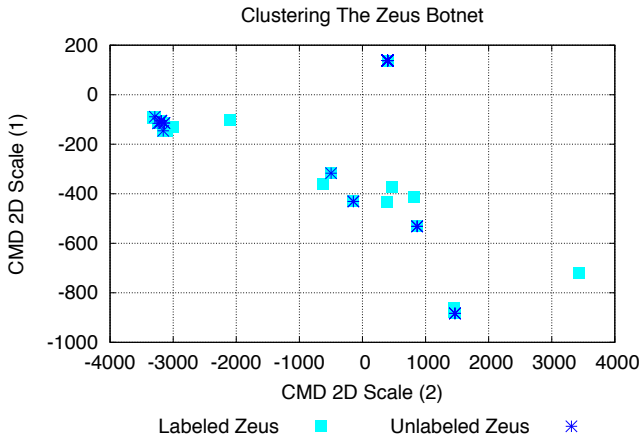


Figure 15: An example of how the Zeus botnet clusters during our experiments. All vectors are in the same network cluster and in two different zone clusters.

and `malwaredomainlist.com` have limited visibility into the botnet, and often produce disjoint blacklists. Notos addresses this problem, by leveraging a limited amount of training data to correctly classify new RRs. During our evaluation set, Notos correctly detected 685 new (previously unknown) Zeus RRs.

3.5.3 Accuracy of the Reputation Function

The first thing that we address in this section is our decision to use a Decision Tree using Logit-Boost strategy (LAD) as the reputation function. Our decision is motivated by the time complexity, the detection results and the precision (true positives over all positives) of the classifier. We compared the LAD classifier to several other statistical classifiers using a typical model selection procedure [25]. LAD was found to provide the most accurate results in the shortest training time for building the reputation function. As we can see from the ROC curve in Figure 11, the LAD classifier exhibits a low false positive rate (FP%) of 0.38% and true positive rate (TP%) of 96.8%. It is was noting that these results were obtained using 10-fold cross-validation, and the detection threshold was set to 0.5. The dataset using for the evaluation contained 10,719 RRs related to 9,530 *known bad* domains. The list

of *known good* domains consisted of the top 500 most popular domains according to Alexa.

We also benchmarked the reputation function on other two datasets containing a larger number of *known good* domain names. We experimented with both the top 10,000 and top 100,000 Alexa domain names. The detection results for these experiments are as follows. When using the top 10,000 Alexa domains, we obtained a true positive rate of 93.6% and a false positive rate of 0.4% (again using 10-fold cross-validation and a detection threshold equal to 0.5). As we can see, these results are not very different from the ones we obtained using only the top 500 Alexa domains. However, when we extended our list of *known good* domains to include the top 100,000 Alexa domain names, we observed a significant decrease in the true positive rate and an increase in the false positives. Specifically, we obtained a TP% of 80.6% and a FP% of 0.6%. We believe this degradation in accuracy may be due to the fact that the top 100,000 Alexa domains include not only professionally run domains and network infrastructures, but also include less good domain names, such as file-sharing, porn-related websites, etc., most of which are not run in a professional way and have disputable reputation[3].

We also wanted to evaluate how well Notos performs, compared to static blacklists. To this end, we performed a number of experiments as follows. Given an instance of Notos trained with data collected up to July 31, 2009, we fed Notos with 250,000 distinct RRs found in DNS traffic we collected on August 1, 2009. We then computed the reputation score for each of these RRs. First, we set the detection threshold to 0.5, and with this threshold we identified 54,790 RRs that had a low reputation (lower than the threshold). These RRs where related to a total of 10,294 distinct domain names (notice that a domain name may map to more than one IP address, and this

[3]A quick analysis of the top 100,000 Alexa domains reported that about 5% of the domains appeared in the SURBL (`www.surbl.org`) blacklist, at a certain point in time. A more rigorous evaluation of these results is left to future work.

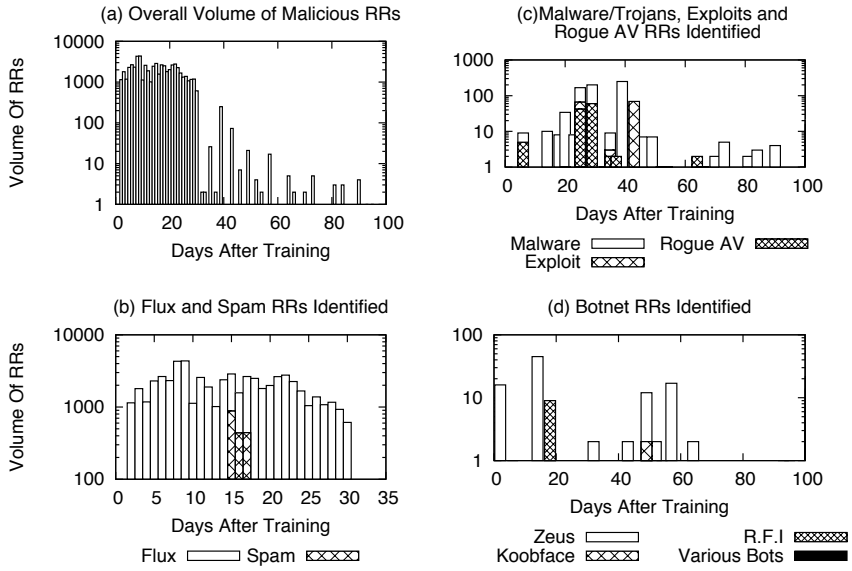


Figure 16: Dates in which various blacklists confirmed that the RRs were malicious after Notos assigned low reputation to them on the 1^{st} of August.

explains the higher number of RRs). Of these 10,294 domains, 7,984 (77.6%) appeared in at least one of the public blacklists we used for comparison (see Section 3.3) within 60 days after August 1, and were therefore confirmed to be malicious. Figure 16(a) reports the number and date in which RRs classified as having low reputation by Notos appeared in the public blacklists. The remaining three plots (Figure 16(b), (c) and (d)), report the same results organized according to the type of malicious domains. In particular, it is worth noting that Notos is able to detect never-before-seen domain names related to the Zeus botnet several days or even weeks before they appeared in any of the public blacklists.

For the remaining 22.4% of the 10,294 domains we considered, we were not able to draw a definitive conclusion. However, we believe many of those domains are involved in some kind of more or less malicious activities. We also noticed that 7,980 or the 7,984 *confirmed bad* domain names were assigned a reputation score lower or equal to 0.15, and that none of the other non-confirmed suspicious domains received a score

Table 6: Sample cases form Zeus domains detected by Notos and the corresponding days that appeared in the public BLs. All evidence information in this table were harvested from zeustracker.abuse.ch.

Domain Name	IP	Date
google-bot004.cn	213.182.197.229	08-15
analf.net	222.186.31.169	08-15
pro-buh.ru	89.108.67.83	08-15
ammdamm.cn	92.241.162.55	08-15
briannazfunz.com	95.205.116.55	08-15
mybank-of.com	59.125.229.73	08-15
oc00co.com	212.117.165.128	08-15
avangadershem.com	195.88.190.29	08-19
securebizccenter.cn	122.70.145.140	08-19
adobe-updating-service.cn	59.125.231.252	09-02
0md.ru	219.152.120.118	09-19
avrev.info	98.126.15.186	09-27
g00glee.cn	218.93.202.100	09-02

lower than this threshold. In practice, this means that an operator who would like to use Notos as a stand-alone dynamic blacklisting system while limiting the false positives to a negligible (or even zero) amount may fine-tune the detection threshold and set it around 0.15.

The plots (b),(c) and (d) in Figure 16 present the number and type of malicious domain names we identified with Notos on the 1^{st} of August. The plots also include the latency between our identification and the actual date that these domains appeared in the various public blacklists. We identified all spam and flux networks (Figure 16(b)) in our dataset within 30 days. Domain names from malicious infrastructure used for malware dropping, rogue AV [18] and various exploits (e.g., iframes [68, 85]) showed up in the public lists up to almost 45 days after we identified them (Figure 16(c)). In plot (d) from Figure 16 we can see the dates that botnet domain names appeared in the various blacklists we used. Using variants of the Zeus botnet as an example, once we have enough domain names labeled as Zeus C&Cs in our knowledge base, Notos can identify new previously unseen domain names as new Zeus C&Cs several

Table 7: Anecdotal cases of malicious domain names detected by Notos and the corresponding days that appeared in the public BLs . [1]: hosts-file.net, [2]: malwareurl.com, [3] siteadvisor.com, [4] virustotal.com, [5] ddanchev.blogspot.com, [6] malwaredomainlist.com

Domain Name	IP	Type	Src	Date
lzwn.in	94.23.198.97	MAL	[1]	08-26
3b9.ru	213.251.176.169	MAL	[2]	08-30
antivirprotect.com	64.40.103.249	RAV	[3]	09-05
1speed.info	212.117.163.165	CWS	[2]	09-05
spy-destroyer.com	67.211.161.44	CWS	[4]	09-05
free-spybot.com	63.243.188.110	RAV	[2]	09-05
a3l.at	89.171.115.10	MAL	[2]	09-09
gidromash.cn	211.95.79.170	BOT	[2]	09-13
iantivirus-pro.com	188.40.52.180	KBF	[5]	09-19
ericwanhouse.cn	220.196.59.19	EXP	[6]	09-22
1165651291.com	212.117.165.126	RAV	[2]	10-06

days before they appear in the various public blacklists. This indicates that the accuracy of the reputation function remains high for the Zeus botnet since we manage to provide enough training data using the information obtained from Zeus tracker [108]. Anecdotal cases from early detection cases using Notos can be found in Tables 6 and 7.

If we are able to provide enough ground truth for a given class of a fraudulent activity (e.g., RBN, flux, botnets), Notos will be able to "learn" this new malicious behavior and will be able to utilize the reputation function to rate any future RRs that resembles that malicious behavior with the appropriate score. Our experiment showed that if the pDNS database contains enough historic DNS data, the reputation function will be able to assign the appropriate score to new RRs with high accuracy and very few false positives.

3.6 Summary

The Domain Name System (DNS) is an essential protocol used by both legitimate Internet applications and cyber attacks. For example, botnets rely on DNS to support agile command and control infrastructures. An effective way to disrupt these attacks is to place malicious domains on a "blocklist" (or "blacklist") or to add a filtering rule in a firewall or network intrusion detection system.

To evade such security countermeasures, attackers have used *DNS agility*, e.g., by using *new* domains daily to evade static blacklists and firewalls. In this Chapter we proposed Notos, a dynamic reputation system for DNS. The premise of this system is that malicious, agile use of DNS has unique characteristics and can be distinguished from legitimate, professionally provisioned DNS services.

Notos uses passive DNS query data and analyzes the network and zone features of domains. It builds models of known legitimate domains and malicious domains, and uses these models to compute a reputation score for a new domain indicative of whether the domain is malicious or legitimate.

We have evaluated Notos in a large ISP's network with DNS traffic from 1.4 million users. Our results show that Notos can identify malicious domains with high accuracy (true positive rate of 96.8%) and low false positive rate (0.38%), and can identify these domains weeks or even months before they appear in public blacklists.

CHAPTER IV

DETECTING MALWARE OUTBREAKS IN THE UPPER LAYERS OF THE DNS HIERARCHY

4.1 Motivation

Over the years Internet miscreants have used the DNS to build malicious network infrastructures. For example, botnets [2, 70, 84] and other types of malicious software make use of domain names to locate their command and control (C&C) servers and communicate with attackers, e.g., to exfiltrate stolen private information, wait for commands to perform attacks on other victim machines, etc. In response to this malicious use of DNS, *static* domain blacklists containing known malware domains have been used by network operators to detect DNS queries originating from malware-infected machines and block their communications with the attackers [54, 59].

Unfortunately, the effectiveness of static domain blacklists are increasingly limited because there are now an overwhelming number of new domain names appearing on the Internet every day and attackers frequently switch to different domains to run their malicious activities, thus making it difficult to keep blacklists up-to-date.

To overcome the limitations of static domain blacklists, we need a detection system that can *dynamically* detect new malware-related domains. This detection system should:

(1) Have *global visibility* into DNS request and response messages related to large DNS zones. This enables "early warning", whereby malware domains can be detected before the corresponding malware infections reach our local networks.

(2) Enable DNS operators to *independently* deploy the system and detect malware-related domains from within their authority zones without the need for data from

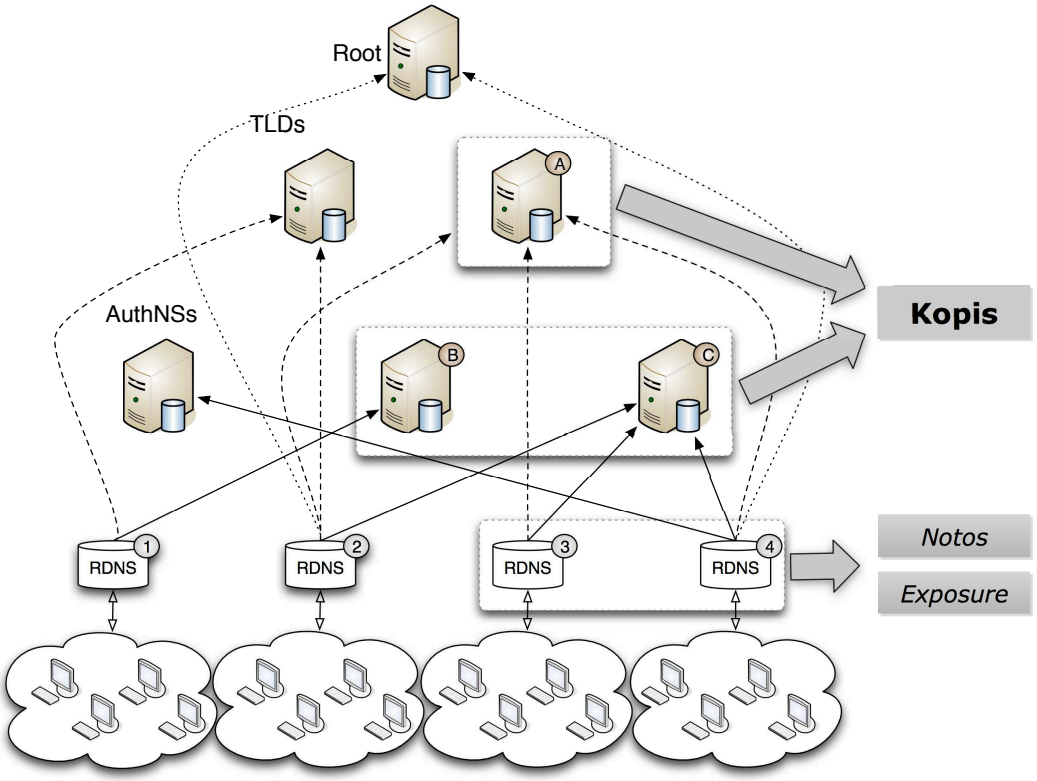


Figure 17: Overview of the levels at which Kopis, Notos, and Exposure perform DNS monitoring.

other networks or other inter-organizational coordination. This enables practical, low-cost, and time-efficient detection and response.

(3) Accurately detect malware-related domains even in the absence of reputation data for the IP address space pointed to by the domains. IP reputation data is often difficult to accumulate and is fragile. This issue may become particularly important as IPv6 is deployed in the near future, due to the more expansive address space.

Recently researchers have proposed two dynamic domain reputation systems, Notos [7] and Exposure [11]. Unfortunately, while the results reported in [7, 11] are promising, neither Notos nor Exposure can meet all the requirements stated above, as Notos and Exposure rely on passive monitoring of recursive DNS (RDNS) traffic.

As shown in Figure 17, they monitor the DNS queries from a (limited) number of RDNS servers (e.g., RDNS `3` and `4`), and have only partial visibility on DNS messages related to large DNS zones. To obtain truly global visibility into DNS traffic related to a given DNS zone, these systems need access to a very large number of RDNS sensors in many diverse locations. This is not easy to achieve in practice in part due to operational costs, privacy concerns related to sharing data across organizational boundaries, and difficulties in establishing and maintaining trust relationships between network operators located in different countries, for example. For the same reasons, Notos and Exposure have not been designed to be independently deployed and run by single DNS operators, because they rely on data sharing among several networks to obtain a meaningful level of visibility into DNS traffic.

On the other hand, monitoring DNS traffic from the upper DNS hierarchy, e.g., at top-level domain (TLD) server `A`, and authoritative name servers (AuthNSs) `B` and `C`, offers visibility on *all* DNS messages related to domains on which `A`, `B`, and `C` have authority or are a point of delegation. For example, assuming `B` is the AuthNS for the `example.com` zone, monitoring the DNS traffic at `B` provides visibility on all DNS messages from all RDNS servers around the Internet that query a domain name under the `example.com` zone.

DNS visibility from TLDs servers (i.e., server `A`) and authoritative DNS servers (i.e., servers `B` and `C`) allows us to analyze *global* query patterns regarding the domains in large DNS zones, thus providing a new "signal" that can be harvested to detect malware-related domain names.

Following this intuition, in this Chapter we present a novel detection system called Kopis, which takes advantage of the global visibility available at the upper levels of the DNS hierarchy to detect malware-related domains. In order for Kopis to satisfy the three requirements outlined above, it needs to deal with a number of new challenges. Most significantly, the higher up we move in the DNS hierarchy, the stronger the

effects of DNS caching [47]. As a consequence, moving up in the hierarchy restricts us to monitoring DNS traffic with a coarser granularity. For example, at the TLD level we will only be able to see a small subset of queries to domains under a certain delegation point due to the effects of the DNS cache.

Kopis works as follows. It analyzes the streams of DNS queries and responses at AuthNS or TLD servers (see Figure 17) from which are extracted statistical features such as the diversity in the network locations of the RDNS servers that query a domain name, the level of "popularity" of the querying RDNS servers (defined in detail in Section 4.3), and the reputation of the IP space into which the domain name resolves. Given a set of known legitimate and known malware-related domains as training data, Kopis builds a statistical classification model that can then predict whether a new domain is malware-related based on observed query resolution patterns.

Our choice of Kopis' statistical features, which we discuss in detail in Section 4.3, is determined by the nature of the information accessible at the upper DNS hierarchy. As a result these features are significantly different from those used by RDNS-based systems such as Notos [7] and Exposure [11]. In particular, we were pleasantly surprised to find that, while Notos and Exposure rely heavily on features based on IP reputation, Kopis' features enabled it to accurately detect malware-related domains even in the absence of IP reputation information. This may become a significant advantage in the near future because the deployment of IPv6 may severely impact the effectiveness of current IP reputation systems due to the substantially larger IP address space that would need to be monitored.

4.1.1 Contributions

To summarize, with Kopis we make the following contributions. We developed a novel approach to detect malware-related domain names at the higher levels of the DNS hierarchy. Our system leverages the global visibility obtained by monitoring

DNS traffic at the upper levels of the DNS hierarchy, and can detect malware-related domains based on DNS resolution patterns.

In addition, Kopis enables DNS operators to *independently* (i.e., without the need of data from other networks) detect malware-domains within their scope of authority, so that action can be taken (in a timely manner) to stop the abuse.

We systematically examined real-world DNS traces from two large AuthNSs and a country-code level TLD server. We performed a rigorous evaluation of our statistical features and identified two new feature families that, unlike previous work, enable Kopis to detect malware domains even when no IP reputation information is available.

We developed a proof-of-concept version of Kopis, and experimented with eight months of real-world data. Our experimental results show that Kopis can achieve high detection rates (e.g., 98.4%) and low false positive rates (e.g., 0.3% or 0.5%). More significantly, Kopis was able to identify previously unknown malware domain names several weeks before they appeared in blacklists or in security forums. In addition, using Kopis we detected the rise of a previously unknown DDoS botnet based in China.

4.2 A Detector for Malware Outbreaks

Kopis monitors streams of DNS queries to and responses from the upper DNS hierarchy, and detects malware domain names based on the observed query/response patterns. An overview of Kopis is shown in Figure 18.

Our system divides the monitored data streams into epochs $\{E_i\}_{i=1..m}$ (currently, an epoch is one day long). At the end of each epoch Kopis summarizes the DNS traffic related to a given domain name d by computing a number of statistical features, such as the diversity of the IP addresses associated with the RDNS servers that queried d, the relative volume of queries from the set of querying RDNS servers, historic information related to the IP space pointed to by d, etc. We defer a detailed

description and motivations regarding the features we measure to Section 4.3. For now, it suffices to consider the *feature computation* module in Figure 18 as a function $\mathcal{F}(d, E_i) = v_d^i$ that maps the DNS traffic in epoch E_i related to d into a feature vector v_d^i.

Kopis operates in two modes: a *training* mode and an *operation* mode. In training mode, Kopis makes use of a *knowledge base* **KB**, which consists of a set of known malware-related and known legitimate domain names (and related resolved IPs) for which the monitored AuthNS and TLD servers are authoritative or a point of delegation. Kopis' *learning module* takes as input the set of feature vectors $\mathbf{V}_{train} = \{v_d^i\}_{i=1..m}, \forall d \in \mathbf{KB}$, which summarizes the query/response *behavior* of each domain in the knowledge base across m days. Each domain in **KB**, and in turn each feature vector in $\mathbf{V}_{train}$, is associated with a label, namely *legitimate* or *malware*. We can therefore use supervised learning techniques [12] to learn a statistical classification model $\mathcal{S}$ of DNS query patterns related to legitimate and malware domains as seen from the upper DNS hierarchy.

In operation mode, Kopis monitors the streams of DNS traffic and, at the end of each epoch E_j, maps each domain $d' \notin \mathbf{KB}$ (i.e., all *unknown* domains) extracted from the query/response streams into a feature vector $v_{d'}^j$. At this point, given a domain d' the statistical classifier $\mathcal{S}$ (see Figure 18) assigns a label $l_{d',j}$ and a confidence score $c(l_{d',j})$, which express whether the query/response patterns observed for d' during epoch E_j resemble either known legitimate or malware behavior, and with what probability. In order to make a final decision about d', Kopis first gathers a series of labels and confidence scores $\mathcal{S}(v_{d'}^j) = \{l_{d',j}, c(l_{d',j})\}, j = t, .., (t+m)$ for m consecutive epochs, where t refers to a given starting epoch E_t. Finally, Kopis computes the average confidence scores $\overline{C}_M = avg_j\{c(l_{d',j})\}$ for the *malware* labels assigned to d' by $\mathcal{S}$ across the m epochs, and an alarm is raised if $\overline{C}_M$ is greater than a threshold θ.

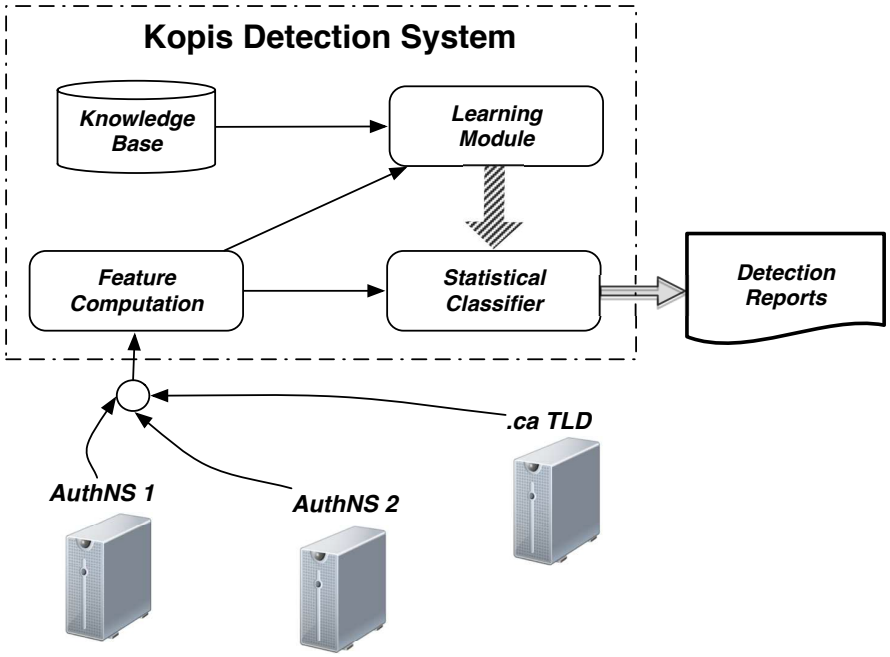


Figure 18: A high-level overview of Kopis.

4.3 Statistical Features

In this section we describe the statistical features that Kopis *extracts* from the monitored DNS traffic. In addition, we will evaluate the merit of each statistical feature family.

4.3.1 Statistical Feature Families

For each DNS query q_j regarding a domain name d and the related DNS response r_j, we first translate it into a tuple $\mathcal{Q}_j(d) = (T_j, R_j, d, IPs_j)$, where T_j identifies the epoch in which the query/response was observed, R_j is the IP address of the machine that initiated the query q_j, d is the queried domain, and IPs_j is the set of resolved IP addresses as reported in the response r_j. It is worth noting that since we are monitoring DNS queries and responses from the upper DNS hierarchy, in some cases the response may be *delegated* to a name server which Kopis does not currently monitor. This is particularly relevant to our TLD-level data feed, since most TLD

servers are *delegation-only*[1]. In all those cases in which the response does not carry the resolved IP addresses, we can derive the *IPs* set by leveraging a passive DNS database [17], or by directly querying the delegated name server.

Given a domain name d and a series of tuples $\mathcal{Q}_j(d), j = 1, .., m$, measured during a certain epoch E_t (i.e., $T_j = E_t, \forall j = 1, .., m$), Kopis extracts the following groups of statistical features:

Requester Diversity (RD) This group of features aims to characterize if the machines (e.g., RDNS servers) that query a given domain name are localized or are globally distributed. In practice, given a domain d and a series of tuples $\{\mathcal{Q}_j(d)\}_{j=1..m}$, we first map the series of requester IP addresses $\{R_j\}_{j=1..m}$ to the BGP prefix, autonomous system (AS) numbers, and country codes (CC) the IP addresses belong to. Then, we compute the distribution of occurrence frequencies of the obtained BGP prefixes (sometimes referred to as classless inter-domain routing (CIDR) prefixes), the AS numbers and CCs.

For each of these three distributions we compute the mean (three features) , standard deviation (three features) and variance (three features). Also, we consider the absolute number of distinct IP addresses (i.e., distinct values of $\{R_j\}_{j=1..m}$), the number of distinct BGP prefixes, AS numbers and CCs (four features in total). Overall, we obtain thirteen statistical features that summarize the diversity of the machines that query a particular domain name, as seen from an AuthNS or TLD server.

The choice of the *RD* features is motivated by the observation that the distribution of the machines on the Internet that query malicious domain names is on average different from the distribution of IP addresses that query legitimate domains. Semi-popular legitimate domain names (i.e., small business or personal sites) will not have a stable diverse population of recursive DNS servers or stubs that will try to

[1]Delegation-only DNS servers are effectively limited to containing `NS` resource records for subdomains, but no actual data beyond its own `SOA` and `NS` records.

systematically contact them. On the other hand popular legitimate domain names (i.e., zone cuts, authoritative name servers, news/blog forums, etc.) will demonstrate a very consistent and very diverse pool of IP addresses looking them up on a daily basis.

Malware-related domain names will have a diverse pool of IP addresses looking them up in a systematic way (i.e., multiple contiguous days). These IP addresses are very likely to have a significant network and geographical diversity simply because with the exception of targeted attacks adversaries will not try to control or restrain the geographical and network distribution of the machines getting compromised by drive-by sites and other social networking techniques. Intuitively, the diversity of the infected population will be different over a given time period, in comparison to that of benign domain names.

For example, Figure 19(a), which is derived from the dataset described in Section 4.4.3, reports the cumulative distribution functions (CDF) of the AS diversity of benign and malware-related domain names. In Figure 19(b) we can see the CDFs from the CC diversity for both classes in our dataset. We note that in both cases the benign domain names (in our dataset) have a bimodal distribution. They either have low or very high diversity. On the other hand, the malware-related domain names cover a larger spectrum of diversities based on the success of the malware distribution mechanisms they use.

Requester Profile (RP) Not all query sources have similar characteristics. Given a query tuple $\mathcal{Q}_j(d) = (T_j, R_j, d, IPs_j)$, the requester's IP address R_j may represent the RDNS server of a large ISP that queries domains on behalf of millions of clients, the RDNS of a smaller organization (e.g., an academic network), or a single end-user machine. We would like to distinguish between such cases, and assign a higher weight to RDNS servers that serve a large client population because a larger network

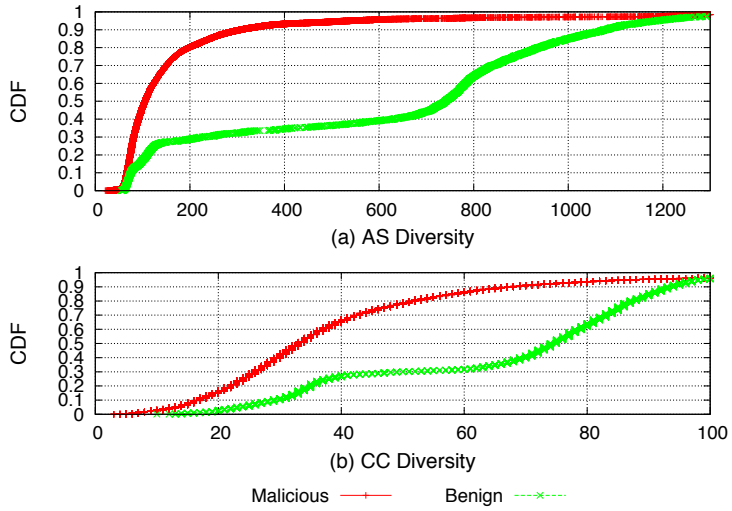


Figure 19: Distribution of AS-diversity (a) and CC-diversity (b) for malware-related and benign domains.

would typically have a larger number of infected machines. While it is not possible to precisely estimate the population behind an RDNS server, because of the effects of caching [47], we approximate the population measure as follows. Without loss of generality, assume we monitor the DNS query/response stream for a large AuthNS that has authority over a set of domains $\mathbf{D}$. Given an epoch E_t, we consider all query tuples $\{\mathcal{Q}_j(d)\}, \forall j, d$ seen during E_t. Let $\mathbf{R}$ be the set of all distinct requester IP addresses in the query tuples. For each IP address $R_k \in \mathbf{R}$, we count the number $c_{t,k}$ of different domain names in $\mathbf{D}$ queried by R_k during E_t. We then define the weight associated to a requester's IP address R_k as $w_{t,k} = \frac{c_{t,k}}{\max_{l=1}^{|\mathbf{R}|} c_{t,l}}$. In practice, we assign a higher weight to requesters that query a large number of domains in $\mathbf{D}$.

Now that we have defined the weights $w_{t,j}$, given a domain name d' we measure its *RP* features as follows:

- Let $\{\mathcal{Q}_i(d')\}_{i=1..h}$ be the set of query tuples related to d' observed during an epoch E_t. Also, let $\mathbf{R}(d')$ be the set of all distinct requester IP addresses in $\{\mathcal{Q}_i(d')\}_{i=1..h}$. For each $R_k \in \mathbf{R}(d')$ we compute the count $c_{t,k}$ as previously

described. Then, given the set $C_t(d') = \{c_{t,k}\}_k$, we compute the average, the biased and unbiased standard deviation[2], and the biased and unbiased variance of the values in $C_t(d')$. It is worth noting that the biased and unbiased estimators of the standard deviation and variance have different values when the cardinality $|C_t(d')|$ is small.

- Similar to the above, for each $R_k \in \mathbf{R}(d')$ we compute the count $c_{t,k}$. Afterwards, we multiply each count by the weight $w_{t-n,k}$ to obtain the set $WC_t(d') = \{c_{t,k} * w_{t-n,k}\}_k$ of weighted counts. It is worth noting that the weights $w_{t-n,k}$ are computed based on historical data about the resolver's IP address collected n epochs (seven days in our experiments) before the epoch E_t. We then compute the average, the biased and unbiased standard deviation, and the biased and unbiased variance of the values in $WC_t(d')$.

The *RD* and *RP* features described above aim to capture the fact that malware-related domains tend to be queried from a diverse set of requesters with a higher weight more often than legitimate domains. An explanation for this expected difference in the requester characteristics is that malware-related domains tend to be queried from a large number of ISP networks, which usually are assigned a high weight. The reason is that ISP networks often offer little or no protection against malware-related software propagation. In addition, the population of machines in ISP networks is usually very large, and therefore the probability that a machine in the ISP network becomes infected by malware is very high. On the other hand, legitimate domains are often queried from both ISP networks and smaller organization networks (having a smaller weight), such as enterprise networks, which are usually better protected against malware and tend to query fewer malware-related domains. As shown in Section 4.4 both set of features can successfully model benign and malware-related

[2]The biased estimator for the standard deviation of a random variable X is defined as $\hat{\sigma} = \sqrt{\sum_{i=1}^{N} \frac{1}{N}(\bar{X}_i - \mu)^2}$, while the unbiased estimator is defined as $\tilde{\sigma} = \sqrt{\sum_{i=1}^{N} \frac{1}{N-1}(\bar{X}_i - \mu)^2}$

domain names.

Resolved-IPs Reputation (IPR) This group of features aims to describe whether, and to what extent, the IP address space pointed to by a given domain has been historically linked with known malicious activities, or known legitimate services. We compute a total of nine features as follows. Given a domain name d and the set of query tuples $\{\mathcal{Q}_j(d)\}_{j=1..h}$ obtained during an epoch E_t, we first consider the overall set of resolved IP addresses $\mathbf{IPs}(d,t) = \cup_{j=1}^{h} IPs_j$ (where IPs_j is an element of the tuple $\mathcal{Q}_j(d)$, as explained above). Let $\mathbf{BGP}(d,t)$ and $\mathbf{AS}(d,t)$ be the set of distinct BGP prefixes and autonomous system numbers to which the IP addresses in $\mathbf{IPs}(d,t)$ belong, respectively. We compute the following groups of features.

- *Malware Evidence*: includes the average number of known malware-related domain names that in the past month (with respect to the epoch E_t) have pointed to each of the IP addresses in $\mathbf{IPs}(d,t)$. Similarly, we compute the average number of known malware-related domains that have pointed to each of the BGP prefixes and AS numbers in $\mathbf{BGP}(d,t)$ and $\mathbf{AS}(d,t)$.

- *SBL Evidence*: much like the malware evidence features, we compute the average number of domains from the Spamhaus Block List [67] that, in the past have pointed to each of the IP addresses, BGP prefixes, and AS numbers in $\mathbf{IPs}(d,t)$, $\mathbf{BGP}(d,t)$, and $\mathbf{AS}(d,t)$, respectively.

- *Whitelist Evidence*: We compute the number of IP addresses in $\mathbf{IPs}(d,t)$ that match IP addresses pointed to by domains in the DNSWL [23] [3] or the top 30 domains according to Alexa [5]. Similarly we compute the number of BGP prefixes in $\mathbf{BGP}(d,t)$ and AS numbers in $\mathbf{AS}(d,t)$ that include IP addresses pointed by domains in DNSWL or the top 30 Alexa domains.

[3]Domain names up to the *LOW* trustworthiness score, where *LOW* trustworthiness score follows the definition by DNSWL [23]. More details can be found at `http://www.dnswl.org/tech`.

The IPR features try to capture whether a certain domain d is related to domain names and IP addresses that have been historically recognized as either malicious or legitimate domains. The intuition is that if d points into IP address space that is known to host lots of malicious activities, it is more likely that d itself is also involved in malicious activities. On the other hand, if d points into a well known, professionally run legitimate network, it is somewhat less likely that d is actually involved in malicious activities.

Discussion: While none of the features used alone may allow Kopis to accurately discriminate between malware-related and legitimate domain names, by combining the features described above we can achieve a high detection rate with low false positives, as shown in Section 4.4.

We would like to emphasize that the features computed by Kopis, particularly the *Requester Diversity* and *Requester Profile* features are completely orthogonal from the statistical features proposed in Notos [7] and Exposure [11], which are heavily based on IP reputation information. Unlike Notos and Exposure, which leverage RDNS-level DNS traffic monitoring, Kopis extracts statistical features specifically chosen to harvest the "malware DNS resolution signal" as seen from the upper DNS levels of the DNS hierarchy, and to cope with the coarser granularity of the DNS traffic observed at the AuthNS and TLD level. In addition, as we show in Section 4.4 that unlike previous work, Kopis is able to detect malware-related domains *even when no IP reputation information is available.*

The *Requester Diversity* and *Requester Profile* features can operate without any historical IP address reputation information. These two feature families can be computed practically and *on-the-fly* at each authoritative or TLD server. The main reason why we identify the six *Resolved-IP Reputation* features is to harvest part of the already established IP reputation in IPv4. This will help the overall system to reduce

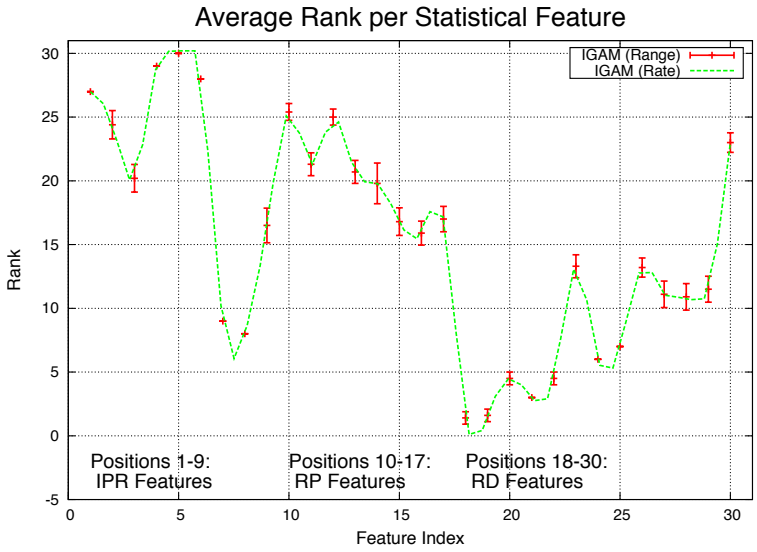


Figure 20: Average ranking for the statistical features based on the information gain criterion. Note that low values of the rank signify higher information gain.

the false positives (FPs) and at the same time maintain a very high true positives (TPs). We will elaborate more in Section 4.4 on the different operational modes of Kopis.

4.3.2 Statistical Feature Analysis

Now that we have defined how Kopis translates streams of DNS query/response into feature vectors, an interesting question is: what is the *merit* of the features we defined? In other words, what features are more discriminating than others? While it is not easy to answer this question, simply because to find the correct answer we would need to consider all combinations of features, the information gain criteria can give us a hint on how each single feature may contribute to improving the classification accuracy.

Assume $l \in \{malicious, legitimate\}$ is a random variable that represents the possible classes for a domain name, and let Y_i be a random variable that represents the i-th feature in the feature vectors Kopis uses to classify the domains. We can

compute the information gain as $IG(Yi) = H(L) - H(L|Y_i)$. In practice, the value of $IG(Y_i)$ tells us how much a certain feature Y_i helps in reducing the uncertainty about the class label l. We can then rank the features according to their information gain. Figure 20 plots the rank of each of the features Kopis uses. Notice that lower values of the rank signify higher information gain. The features are identified by an ID number, and grouped as reported in the text above the x axes. As we can see, several features in the *Requester Diversity* group have high information gain (low rank number in the graph). Also, some of the *Requester Profiling* features offer more information gain than some of the *Resolved-IPs Reputation* features.

Kopis is a data-driven system based on machine learning algorithms that generate accurate models that are not necessarily interpretable. For example the nearest neighbor classifier gives a result without any further information about the contribution of each feature. For that reason we used a set of other state of the art machine learning methods that rank the merit of each feature. It is possible though that two features might have high merit while being correlated. For that reason we also use techniques that can identify dependencies between features.

Information gain is a very widely accepted measure for ranking the merit of features in classification. We tried three methods based on the information gain illustrated in Figure 20.

The gain ratio attribute evaluation method (GRAM) evaluates the worth of an attribute by measuring the gain ratio with respect to the class. In other words GRAM is:

$$GRAM(X, Y_i) = \frac{H(X) - H(X|Y_i)}{H(Y_i)},$$

where H is the entropy, X is the class and Y_i is the i^{th} feature in our feature space.

The information gain attribute evaluation method (IGAM) evaluates the merit of an attribute by measuring the information gain with respect to the class. In other words IGAM is:

$$IGAM(X, Y_i) = H(X) - H(X|Y_i),$$

where H, X and Y_i follow the same definition as in GRAM.

The last feature evaluation method is the symmetrical uncertainty attribute evaluation method (SUAM) that evaluates the worth of an attribute by measuring the symmetrical uncertainty with respect to the class. In other words SUAM is:

$$SUAM(X, Y_i) = 2 \times \frac{H(X)-H(X|Y_i)}{H(X)+H(Y_i)},$$

where H, X and Y_i follow the same definition as in GRAM.

In Figure 20 we can observe the average ranking of each feature. We should note that the ranking is made in such order so the most important feature will have rank zero. We observe that the RDATA reputation features (1-9) do not dominate in overall merit. We see that both the RDNS profiling features (10-17) and the RDNS diversity features (18-30) are ranked significantly higher. This implies that our decision of selecting the two RDNS-based feature families outside the traditional DNS reputation features looks to be very promising towards our effort of discriminating the benign and malicious domain names. The methods do not necessarily give the same ranking for the features. For example the top three features for the GRAM and IGAM methods are the 18,19 and 21 while for the SUAM are the 7,18 and 19. Closer analysis of the results shows that the sets of top k features from every method might be different in no more than two positions in the overall ranking.

Finally, if we take the three sets of the top 19 features from each method we will end up having the following 18 features in common, which we define as the minimum set of statistical features that Kopis could operate on.

- **RD feature set (12):** which include the average, variance and standard deviation of the resolution request counts per CC, AS and CIDR in a day. Furthermore, we will select the count of the RDNS servers that resolve the domain

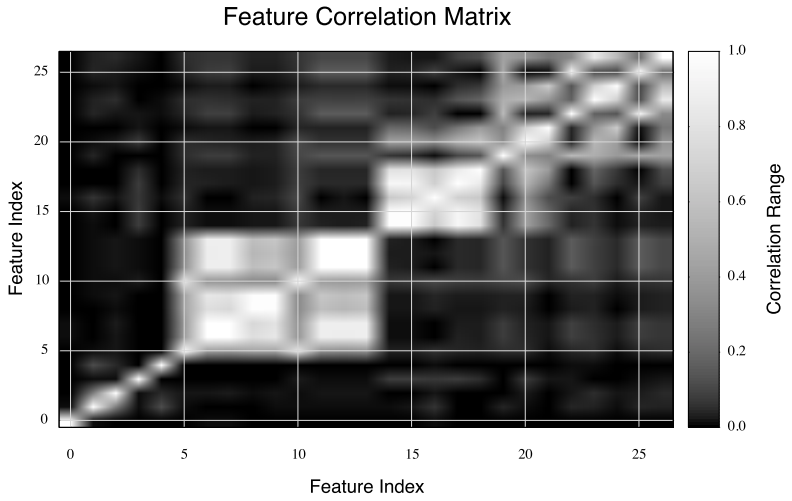


Figure 21: The correlation matrix between features obtained using unsupervised PCA.

name in a day along with the diversity count of the different CCs and CIDRs that resolve each domain name in a day.

- **RP feature set (4):** which include the average, variance, standard deviation and the population standard deviation of the RVF features.

- **IPR feature set (2):** which include the white-listed IP address count and the number of IP addresses that historically were marked as malicious in the same CIDR as the IP addresses mapped with the domain name .

The next problem we need to address in this section is to ensure that we do not have any redundant features in our dataset. A simple test is the correlation coefficient between the features. If there is any linear dependency between two features then the magnitude of the correlation coefficient is one. On the other extreme if they are independent then it is zero. In Figure 21, we can see the correlation matrix for all features in our dataset after using principal component analysis (PCA). A closer look

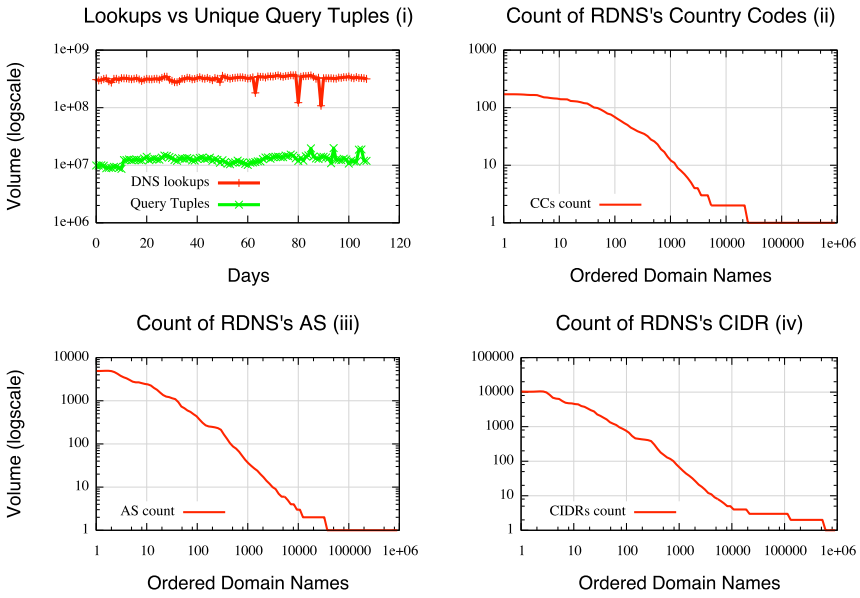


Figure 22: General observations from the datasets. Plot (i) shows the difference between the raw lookup volume vs. the query tuples that Kopis uses over a period of 107 days. Plots (ii), (iii) and (iv) show the number of unique CCs, ASs and CIDRs (in which the RDNSs resides) for each domain name that was looked up during one day.

at the figure shows that there is strong correlation between features 5-9 (Black and SBL features from the IPR feature family) and 10-14 (RP features).

4.4 Evaluation

In this section, we report the results of our evaluation of Kopis. First, we describe how we collected our datasets and the related ground truth. We then present results regarding the detection accuracy of Kopis for authoritative NS- and TLD-level deployments. Finally, we present a case study regarding how Kopis was able to discover a previously unknown DDoS botnet based in China.

4.4.1 Datasets

Our datasets were composed of the DNS traffic obtained from two major domain name registrars between the dates of 01-01-2010 and 08-31-2010 and a country code

top level domain (`.ca`) between the dates of 08-26-2010 and 10-18-2010. In the case of the two domain name registrars we were also able to observe the answers returned to the requester of each resolution. Therefore, it is easy for us to identify the IP addresses for the `A-type` of DNS query traffic. In the case of the TLD we obtained data only for 52 days and had to passively reconstruct the IP addresses corresponding to the `A-type` of lookups observed.

An interesting problem arises when we work with the large data volume from major authorities and the `.ca` TLD servers. According to a sample monitoring period of 107 days we can see from Figure 22 (i) that the daily number of lookups to the authorities was on average 321 million. This was a significant problem since it would be hard to process such a volume of raw data, especially if the temporal information from these daily observations were important for the final detection process. On the same set of raw data we used a data reduction process that maintained only the query tuples (as defined in Section 4.3). This reduced the daily observations, as we can observe from Figure 22 (i), to a daily average of 12,583,723 **unique** query tuples. The signal that we missed with this reduction was the absolute lookup volume of each query tuple in the raw data. Additionally, we missed all time sensitive information regarding the periods within a day that each query tuple was looked up. As we will see in the following sections, this reduction does not affect Kopis' ability to model the profile of benign and malware-related domains.

Figures 22 (ii), (iii) and (iv) report the number of CIDR (i.e., BGP prefixes), Autonomous Systems (AS), Country Code (CC), respectively, for the RDNSs (or requesters) that looked up each domain name every day. The domains are sorted based on counts of ASs, CCs and CIDRs corresponding to the RDNSs that look them up (from left to right with the leftmost having the largest count). We observe that roughly the first 100,000 domain names were the only domains that exhibit any diversity among the requesters that looked them up. We can also observe that the

first 10,000 domain names are those that have some significant diversity. In particular only the first 10,000 domain names were looked up by at least five CIDRs, or five ASs or two different CCs. In other words, the remaining domains were looked up from very few RDNSs, typically in small sets of networks and a small number of countries. Using this observation we created statistical vectors only for domain names in the sets of the 100,000 most diverse domains from the point of view of the RDNS's CC, AS and CIDR.

4.4.2 Obtaining the Ground Truth

We collected more than eight months of DNS traffic from two DNS authorities and the `.ca` TLD. All query tuples derived from these DNS authorities were stored daily and indexed in a relational database. Due to some monitoring problems we missed traffic from 3 days in January, 9 days in March and 6 days in June 2010.

Some of our statistical features require us to map each observed IP address to the related CIDR (or BGP prefix) AS number and country code (Section 4.3). To this end, we leveraged Team CYMRU's IP-to-ASN mapping [88].

Kopis' knowledge base contained malware information from two malware feeds collected since March 2009. We also collected public blacklisting information from various publicly available services (e.g., Malwaredomains [54], Zeus tracker [108]). Furthermore, we collected information regarding domain names residing in benign networks from DNSWL [23] and also the address space from the top 30 Alexa [5] domains verified using the assistance of the Dihe's IP address index browser [21]. Overall, we were able to label 225,429 unique RRs that correspond to 28,915 unique domain names. From those we had 1,598 domain names labeled as legitimate and 27,317 domain names labeled as malware-related.

All collected information was placed in a table with first and last seen timestamps. This was important since we computed all IPR features for day n based only on data

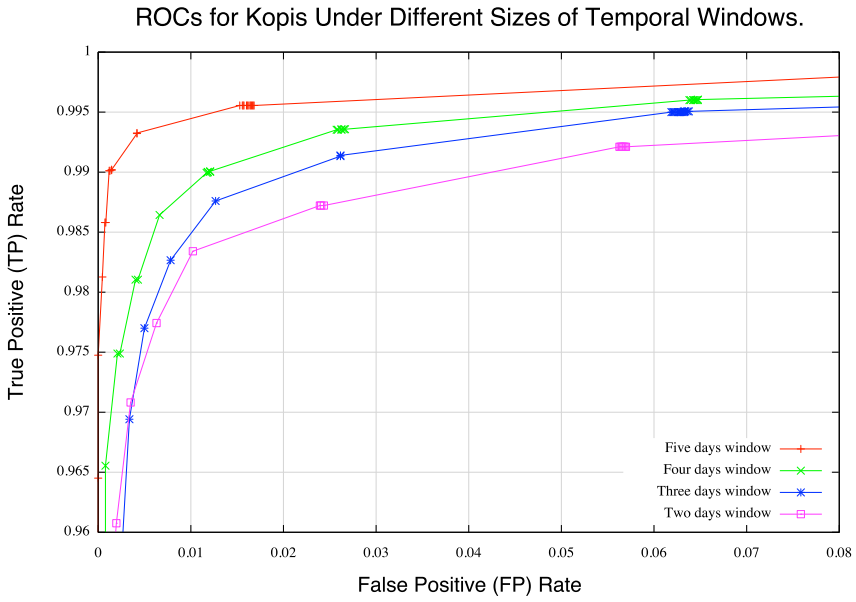


Figure 23: ROCs from datasets with different sizes assembled from different time windows.

we had until day n. Finally, we should note that we labeled all the data based on black-listing and white-listing information collected until October 31^{st} 2010.

4.4.3 Model Selection

As described in Section 4.2, Kopis uses a machine learning algorithm to build a detector based on the statistical profiles of resolution patterns of legitimate and malware-related domains. As with any machine-learning task, it is important to select the appropriate model and important parameters. For Kopis, we need to identify the minimal observation window of historic data necessary for training. The observation window here is the number of epochs from which we assemble the training dataset.

In Figure 23, we see the detection results from four different observation windows. The ROCs in Figure 23 were computed using 10-fold cross validation. The classifier that produced these results was a random forest [82] (RF) classifier under a two, three, four and five day training window. The selection of the RF classifier was

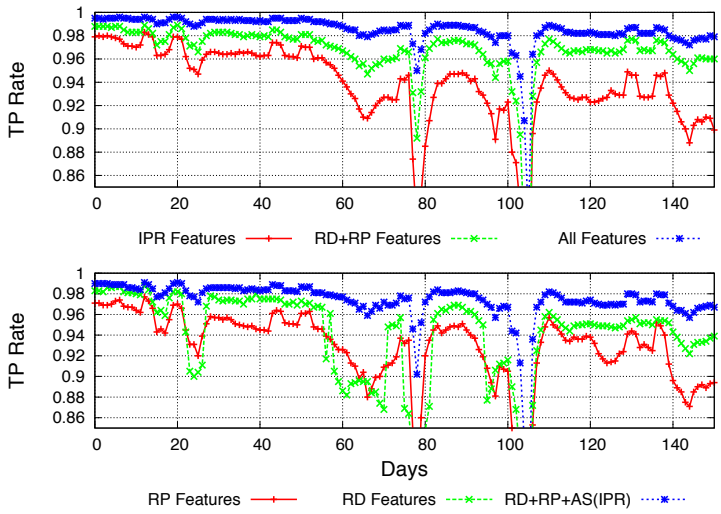


Figure 24: The distribution of TP_{rate} for combination of features and features families in comparison with Kopis observed detection accuracy.

made using a model selection process [25], a common method used in the machine learning community, which identified the most accurate classifier that could model our dataset. Besides the RF, during model selection we also experimented with Naive Bayes [45], k-nearest neighbors [4] (IBK), Support Vector Machines [15], MLP Neural Network [25] and Random (Tree) Committee [25] (RC) classifiers. The best detection results reported during the model selection were from the RF classifier. Specifically, the RF classifier achieved a $TP_{rate} = 98.4\%$ and a $FP_{rate} = 0.3\%$ using a five day observation window. When we increased the observation window beyond the mark of five days we did not see a significant improvement in the detection results.

We should note that this parameter and model methodology should be used every time Kopis is being deployed in a new AuthNS or TLD server because the characteristics of the domains, and hence the resolution patterns, may vary in different AuthNS and TLD servers, and different patterns or profiles may best fit different parameter values and classifiers.

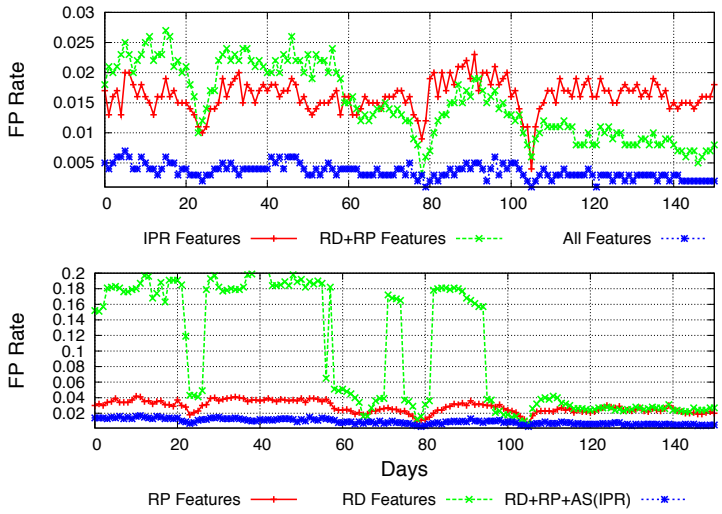


Figure 25: The distribution of FP_{rate} for combinations of features and features families in comparison with Kopis observed detection accuracy.

4.4.4 Overall Detection Performance

In order to evaluate the detection performance of Kopis and in particular the validity and strength of its statistical features and classification model, we conducted a *long-term* experiment with five months of data. We used 150 different datasets created over a period of 155 days (first 15 days for bootstrap). These datasets were composed by using a fifteen-day sliding window with a one-day step (i.e., two consecutive windows overlap by 14 days). We then used 10-fold cross validation[4] to obtain the FP_{rates} and TP_{rates} from every dataset. We picked three classification algorithms, namely, RF, RC, and IBK, which performed best in the model selection process (described in Section 4.4.3) because we wanted to use their detection rates during the long-term experiment.

In Figure 24 and Figure 25 we observe the distribution of the TP_{rates} and FP_{rates} for the RF classifier over the entire evaluation period. The average, minimum and

[4]To avoid overfitting our dataset we report the evaluation results using 10-fold cross validation that implies that 90% of dataset is used for training and 10% for testing — in each of the 10 folds. This technique is known [41] to yield a fair estimation of classification performance over a dataset.

maximum FP_{rates} for the RF were 0.5% (8 domains), 0.2% (3 domains) and 1.1% (18 domains), respectively, while the average, minimum and maximum TP_{rates} were 99.1% (27,072 domains), 98.1% (27.071 domains) and 99.8% (27,262 domains), respectively. The RF classifier's FP_{rates} were almost consistently around 0.6% or less. The TP_{rate} of the RF classifier, with the exception of six days, was above 96% and typically in the range of 98%. With the IBK classifier being the exception, the RF and RC classifiers had similar longterm detection accuracy. This experiment showed that Kopis overall has a very high TP_{rate} and very low FP_{rate} against all new and previously unclassified malware-related domains.

As described in Section 4.3, we define three main types of features. Next we show how Kopis would operate if trained on datasets assembled by features from each family, first separately and then combined. To derive the results from the experiments, we used as input the 150 datasets created in the previously described longterm evaluation mode. Then, for each one of these 150 datasets, we isolated the features from the RD, RP and IPR feature families into three additional types of datasets. In Figure 24 and Figure 25 we present the longterm detection rates obtained using 10-fold cross validation of these three different types of datasets. Additionally, we present the detection results from:

- The combination of RP and RD features (`RD+RP Features`).
- The combination of RD, RP and the features from the IPR feature family that describe the Autonomous System properties of the IP address that each domain name d points at (`RD+RP+IRP(AS) Features`).
- The detection results from the combination of all features (`All Features`).

The longterm FP_{rates} and TP_{rates} in Figure 24 and Figure 25 respectively, we show the detection accuracies from each different feature set. One may tend to think that the IPR (IP reputation) features hold a significantly stronger classification signal than

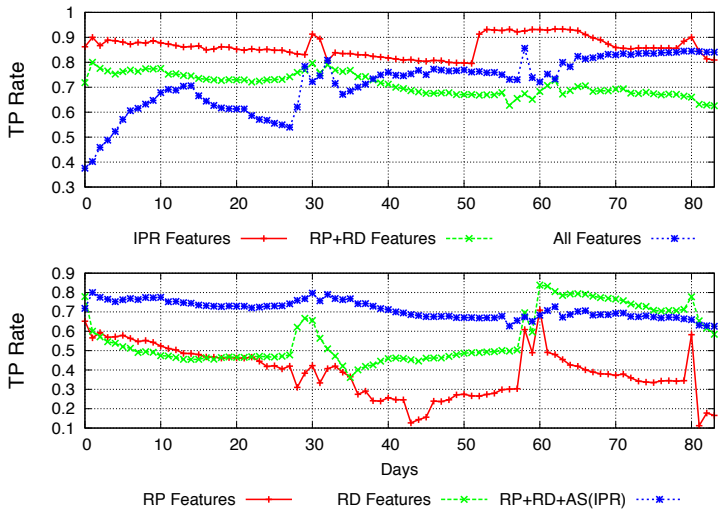


Figure 26: TP_{rates} for different observation periods using an 80/20 train/test dataset split.

the combination of RD and RP features, mainly because there are many resources that currently contribute to the quantification and improvement of IP reputation (i.e., spam block lists, malware analysis, dynamic DNS reputation etc.). However, Figure 24 and Figure 25 show that with respect to both the FP_{rates} and TP_{rates}, the combination of the RD and RP sets of features performs almost equally to the IPR features used in isolation from the remaining features. At the same time, using all features performs much better than using each single feature subset in isolation. This shows that the combination of the RP and RD features contribute significantly to the overall classification accuracy and can enable the correct classification of domains in environments where IP reputation is absent or in cases where we cannot reliably compute IP reputation features "on-the-fly" (e.g., in some TLD-level deployments).

4.4.5 New and Previously Unclassified Domains

While the experiments described in Section 4.4.4 showed that Kopis can achieve very good overall detection accuracy, we also wanted to evaluate the "real-world value" of Kopis, and in particular its ability to detect new and previously unclassified

malware domains. To this end, we conducted a set of experiments in which we trained Kopis based on one month of labeled data from which we randomly excluded 20% of both benign and malware-related domains (i.e., we assumed that we did not know anything about these domain names during training). This excluded 997 benign and 4,792 malware-related unique, deduplicated domain names from the training datasets. Then we used the next three weeks of data as an evaluation dataset, which contained the domains excluded from the training set mentioned above, as well as all other newly seen domain names. In other words, the classification model learned using the training data was not provided with any knowledge whatsoever about the domains in the evaluation dataset.

We then classified the domains in the evaluation dataset, with the assistance of a Random Forest classifier, as we already discussed in Section 4.2.

We used a training period of 30 consecutive days and a testing period of $m = 21$ days immediately following the training period. The detection threshold θ was set to 0.9 to obtain a good operational trade-off between false positives and detection rate. Our primary reasoning behind setting the threshold θ to 0.9 was to keep the FP_{rates} as low as possible so that an operator would only have to deal with a very small number of FPs on a daily basis. We repeated this evaluation four times during different months within our eight months of traffic monitoring.

In Figure 26 and Figure 27, we can see the results of these experiments. From left to right, we can see the evaluation on 21 days of traffic in February, March, May and June of 2010. We trained the system based on one month of traffic from January, February, March and May 2010, respectively. We chose these months because we had continuous daily observations (i.e., no data gaps) from both training and testing datasets. As in the longterm 10-fold evaluation, we performed the experiments using six different datasets obtained using different feature subsets.

We present the results in the same way as in Section 4.4.4. When we used all

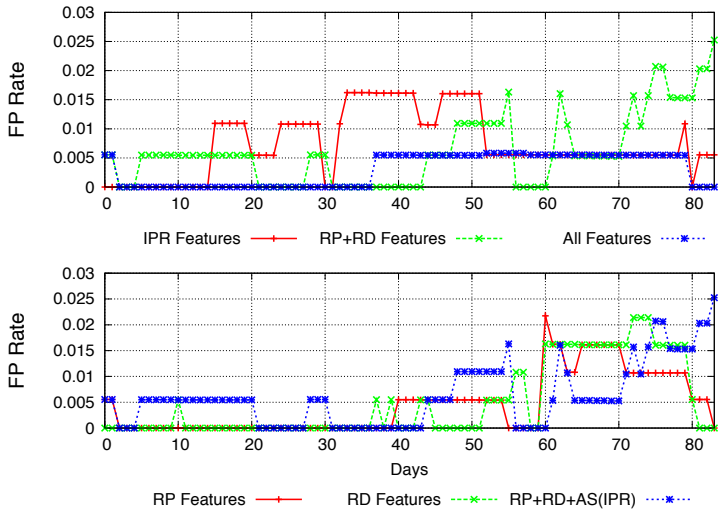


Figure 27: FP_{rates} for different observation periods using an 80/20 train/test dataset split.

features we observed the average FP_{rates} was 0.53% ($\sim$ two domains), while the average TP_{rates} was 73.62% (3,528 domain names). For the `RP+RD Features` and `IPR Features` the average FP_{rates} were 0.54% ($\sim$ two domains) and 0.79% ($\sim$ two domains), respectively; while the average TP_{rates} were 69.19% (3,315 domain names) and 87.25% (4,181 domain names), respectively. The `RP+RD+AS(IPR) Features`, gave average $FP_{rates} = 0.66\%$ (or $\sim$ two domain names) and average $TP_{rates} = 65.05\%$ (or 3,117 domain names).

When we used the combination of all features we see that for the first 42 days of evaluation (February and March of 2010) Kopis had a virtually zero FP_{rates} and an average $TP_{rates} = 68\%$. In the following 42 days of evaluation, Kopis, had better TP_{rates} but with some extra false positives, always below 0.5%. Investigating the nature of the false positives, we observed that the domain names responsible are related to BitTorrent services, on-demand web-TV services and what appeared to be on-line gaming sites. We suspect that the main reason why these domains cause false

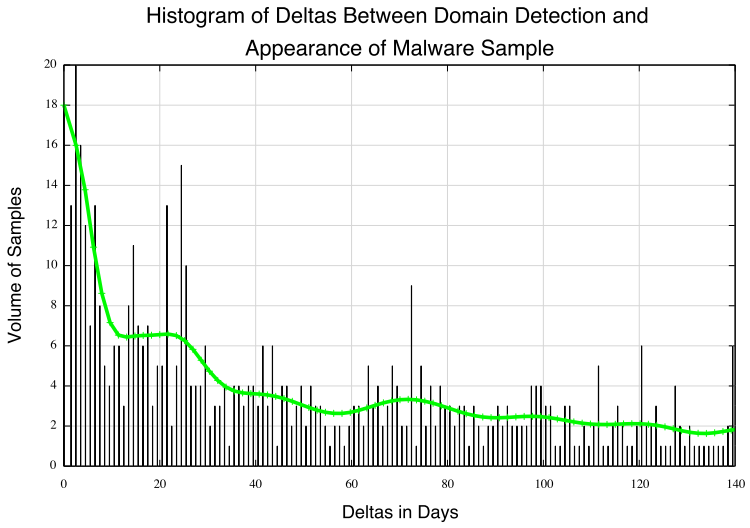


Figure 28: Kopis early detection results. The deltas in days between the Kopis classification dates and the date we've received a corresponding malware sample for the domain name.

positives is because the population of similar legitimate services was insufficiently represented during training, and therefore, the RF classifier failed to learn this behavior as being legitimate in training.

This experiment showed that Kopis — with all features used — can detect new and previously unclassified domains with an average TP_{rate} of 73.62% and average FP_{rate} of 0.53%. Although this is worse than the overall detection performance reported in Section 4.4.4, it is actually a good result considering that Kopis has no knowledge of the domains in the testing dataset. It implies that Kopis has good "real-world value" thanks to its ability to detect new, previously unseen attacks is at a premium.

Figure 28 shows the difference in days between the time that Kopis identifies a *true positive* domain as being malware-related, and the day we first obtained the malware sample associated with the malware-related domain from our malware feed. To perform this measurement, we used malware from a commercial malware feed with volume between 400 MB to 2 GB of malware samples every day. Additionally, we

used malware captured from two corporate networks. As we can see, Kopis was able to identify domain names on the rise **even before** a corresponding malware sample is accessible to the security community. This result shows that Kopis can provide the ability to registrars and TLD operators to preemptively block or take down malware related domains and remove botnets from the Internet before they become a large security threat.

4.4.6 Canadian TLD

Thus far, the experiments we have reported were all using data available at AuthNSs. A TLD server is one level above AuthNS servers in the DNS hierarchy, and as such, it has a greater global visibility but with less granular data on DNS resolution behaviors. In this section we report our experiments of Kopis at the TLD level.

We evaluated Kopis on query data obtained from the Canadian TLD. We used the same evaluation method introduced in Section 4.4.5 but with different training window sizes, testing epochs and classification thresholds. Before we describe the results, we should note that all TLD traffic needs passive reconstruction of the query data to identify the IPs addresses in the `A-type` resource records. We used a passive DNS database composed of data from four ISP sensors and the passive DNS database from SIE [17]. The Canadian TLD's traffic was harvested from SIE [17] (channel three).

Unfortunately, due to the fact that we obtained traffic from only 52 days (2010-08-26 until 2010-10-18) we had to use a smaller training epoch of 14 days (instead of one month). We evaluated Kopis using the RF classifier, 14 consecutive days as the training epoch, 14 days following the training epoch as the evaluation epoch, and setting the threshold $\theta = 0.9$. Two sequential training epochs had seven days in common. The exact training epochs were *08-27* to *09-11*, *09-04* to *09-18*, *09-11* to *09-25* and *09-18* to *10-02* while the corresponding evaluation epochs were *09-12*

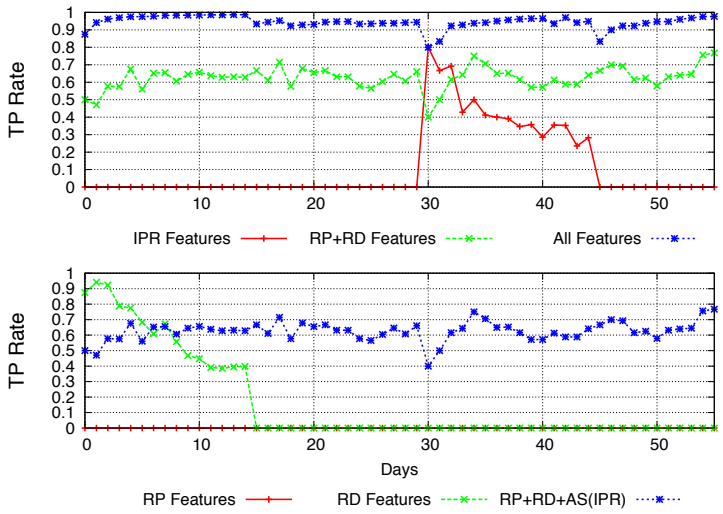


Figure 29: TP_{rates} achieved during evaluation of traffic obtained from `.ca` TLD.

to *09-26*, *09-19* to *10-03*, *09-26* to *10-10* and *10-03* to *10-17*, respectively. Without changing the data labeling methodology, we assembled a dataset with 2,199 malware related and 1,018 benign unique deduplicated domain names.

In Figure 29 and Figure 30, we can see the results of this experiment. As with the experiments in Section 4.4.5, we evaluated Kopis in six modes, using as threshold $\theta = 0.5$. We should note here that the evaluation of the `RD+RP Features` reflects the evaluation mode with datasets that were composed only by the combination of RD and RP features. Such dataset can be extracted directly from data readily available at a TLD server (in other words, the `RD+RP Features` is the most "efficient" mode that Kopis can operate in and can be computed on the fly at a TLD server).

When we used all features we observed the average FP_{rates} was 0.52% ($\sim$ six domain names), while the average TP_{rates} was 94.68% (2,082 domain names). For the `RP+RD Features` and `IPR Features` the average FP_{rates} were 3.18% ($\sim$ 33 domain names) and 0.36% ($\sim$ four domain names), respectively; while the average TP_{rates} were 63.63% (1,399 domain names) and 10.84% (238 domain names), respectively. The `RP+RD+AS(IPR) Features`, gave the average $FP_{rates} = 1.03\%$ (or ten domain

names) and average $TP_{rates} = 78.95\%$ (or 1,736 domain names).

During the `RP+RD Features` evaluation, we observed that the average TP_{rates} reached 63.63% while the average FP_{rates} were in the range of 3.18%. These were very promising results despite the relatively high FP_{rates} because we can operate Kopis using a sequential classification mode, starting with `RP+RD Features` followed by `All Features`. Kopis in this "in-series" classification mode can achieve a good balance of efficiency and accuracy.

More specifically, at the first step in the sequential process, Kopis is a "coarse filter" that operates in `RP+RD Features` with only the RP and RD statistical features and threshold $\theta = 0.5$. Any domain name that passes this filter (i.e., with a "malware-related" label) then requires additional feature computation, i.e., reconstructing the resolved IP address records, and further classification at the next step in the sequential process. On the other hand, domains that are dropped by this filter (i.e., with a "legitimate" label) are no longer analyzed by Kopis. Thus, the first step filter is essentially a data reduction tool, and the sequential classification process is a way to delay the expensive computation until the data volume is reduced. This technique is very important at the TLD level given the potentially huge volume of data.

In our experiments Kopis operating at the first step with `RP+RD Features` (and threshold $\theta = 0.5$) yielded an average data reduction rate[5] of 87.95% on the original dataset. After this reduction, at the second step, we evaluated Kopis on the (remaining) dataset using all features, and keeping the same threshold $\theta = 0.5$. The average FP_{rates} reported at this step by Kopis were zero while the average TP_{rates} were 94.44%. The overall FP_{rates} and TP_{rates} for this "in-series" mode were zero and 60.09% (1,321 domain names), respectively.

At this point we should note that the threshold θ was set again with the intention

[5]We define the reduction rate as follows: $1 - \frac{TP_{malware}+FP_{malware}}{ALL}$, where $TP_{malware}$ is the true positives for the malware-related class, $FP_{malware}$ is the mis-classified as malware-related benign domain names and ALL all as the domain names in the evaluation dataset.

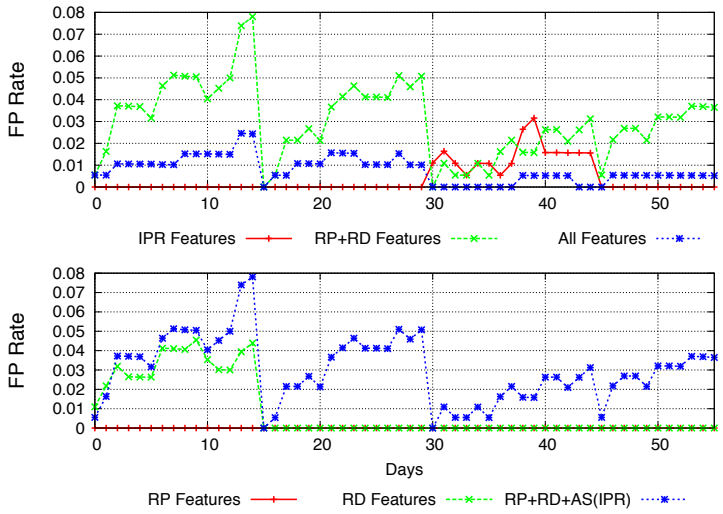


Figure 30: FP_{rates} achieved during evaluation of traffic obtained from `.ca` TLD.

to have the FP_{rates} as close to 1.0% as possible but also not to sacrifice much of the TP_{rate} produced from the first classification process in the "in-series" mode. As we saw previously, even when we had some FPs created by the `RP+RD Features` (the first classification process in the "in-series" mode), the combination of statistical features in the second "in-series" mode was able to prune away these FPs. An operator may choose to lower the threshold θ even more and have as an immediate effect, the increase of domain names that will be forwarded to the second "in-series" classification process, with a potential increase in the overall TP_{rate} and FP_{rates}. The experiments in this section showed that by using an "in-series" classification process where different steps can use different (sub)sets of features and thresholds, Kopis can achieve a good balance of detection performance and operation efficiency at the TLD level.

4.4.7 DDos Botnet Originated in China

As discussed in Section 4.1, Kopis was designed to have global visibility so that it can detect domains associated with malware activities running in an uncooperative country or networks before the attacks propagate to networks that it protects. In this

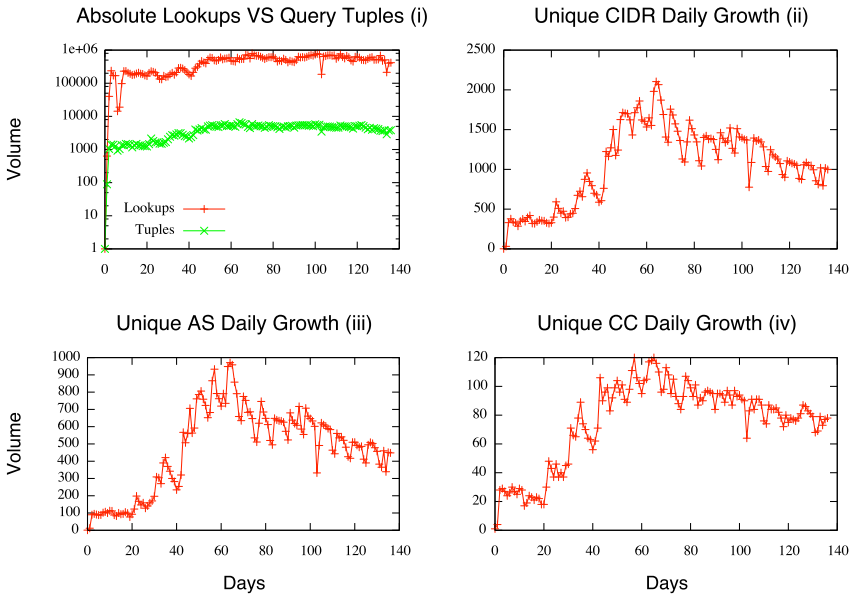


Figure 31: Various growth trends for the DDoS botnet. Day zero is *03-20-2010*.

section, we report a case study to demonstrate Kopis's global detection capability.

Kopis was able to identify a commercial DDoS botnet in the first few weeks of its propagation in China and well before it began propagating within other countries, including the US. We alerted the security community, and the botnet was finally removed from the Internet in the middle of September 2010. Next we provide some intuition behind this discovery and why Kopis was able to detect this threat early.

This DDoS botnet was controlled through 18 domain names, all of which were registered by the attacker under the same authority (although with different 2LDs). Kopis was deployed at the AuthNS server and was able to observe resolution requests to these domains (even when the infected machines were initially not in the US) and classify them as malware-related because their resolution patterns fit the profiles of known malware domains in its knowledge base.

These domain names were linked with six IP addresses located in the following autonomous systems: 14745 (US), two in 4837 (CN), 37943 (CN) and two in 4134

(CN), throughout the lifetime of the botnet. We show the difference between the absolute DNS lookups versus the daily volume of unique query tuples in Figure 31 (i). The average lookup volume every day was 438,471 with average de-duplicated query tuples in the range of 3,883. Despite this significant data reduction, Kopis was still able to track and identify this emerging threat. In Figures 31 (ii), (iii) and (iv), we can see the daily growth of unique CIDRs, AS and CCs related to the RDNSs that queried the domain names used in the botnet.

An interesting observation can be made from Figure 32. In this figure we can see the daily lookup volume for the domain names of this botnet. Instantly we can see that the first big infection happened in Chinese networks in a relatively short period of time (in the first 2-3 days). After this initial infection, a number of machines from several other countries were also infected but nowhere close to the volume of the infected population in the Chinese networks. As an example we can see in Figure 32 that the first time more than 1,000 daily lookups were observed from the United States was more than 20 days after the botnet was launched. Also, other countries such as Poland and Thailand had the first infection 21 and 25 days after the botnet were lunched. Furthermore, large countries such as Italy, Spain and India reached the 100 daily lookup threshold 15 days later than the start of this botnet. Clearly, for countries like Poland and Thailand (and even Italy, Spain and India to a large extent) localized DNS reputation techniques could not have been able to observe a resolution request (or a strong enough signal) for any of the domain names related to this botnet, until the botnet had reached global scale, which was several weeks after it was launched. Figure 33 shows the volume of samples correlated with this botnet as they appeared in our malware feeds. We observe that the first malware sample related to this botnet appeared two months after the botnet became active.

To demonstrate the contribution of each feature family towards the identification

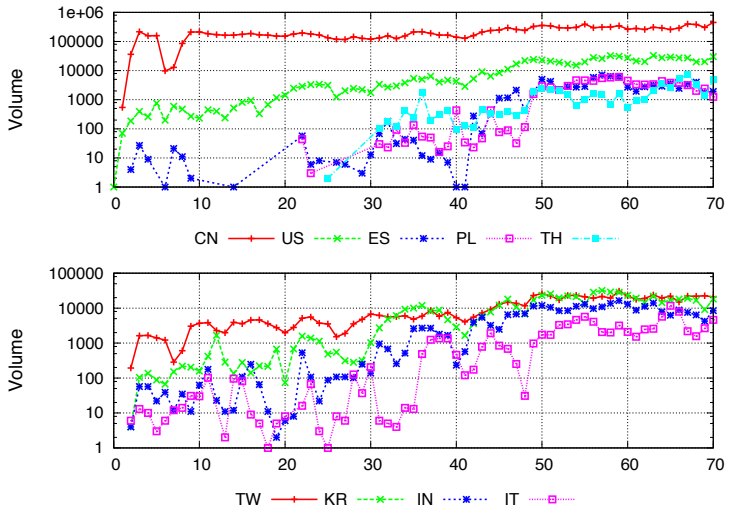


Figure 32: A snapshot from the first 70 days of the botnet's growth with respect to the country code-based resolution attempts for the DDoS botnet's domain names. Day zero is *03-20-2010*.

of the domain names that were part of this botnet we conducted the following experiment. We trained Kopis with 30 days of data before the 5^{th} of May 2010. Then we computed vectors for all the domain names that were part of the botnet. We computed one vector every day for each domain name based on the information we had on the domain name and IP address up until that day. We classified each vector against four trained classifiers with the following set of features; `All Features`, `RD+RP Features`, `IPR Features`, and `RD+RP+AS(IPR) Features`. We then marked the first day that each classifier detected a domain name as malware-related, while setting the threshold $\theta = 0.9$. By doing so we identified the earliest day that the classifier would have detected the domain name without human forensic analysis on the results. The detection results from this experiment can be found in Table 8.

What the results show is that only the combination of all features can detect all the domain names until the end of August. On the other hand the IPR and the combination of RD+RP features detected more than half of the domain names by the middle of July, when the botnet was in its peak. We should also note that in

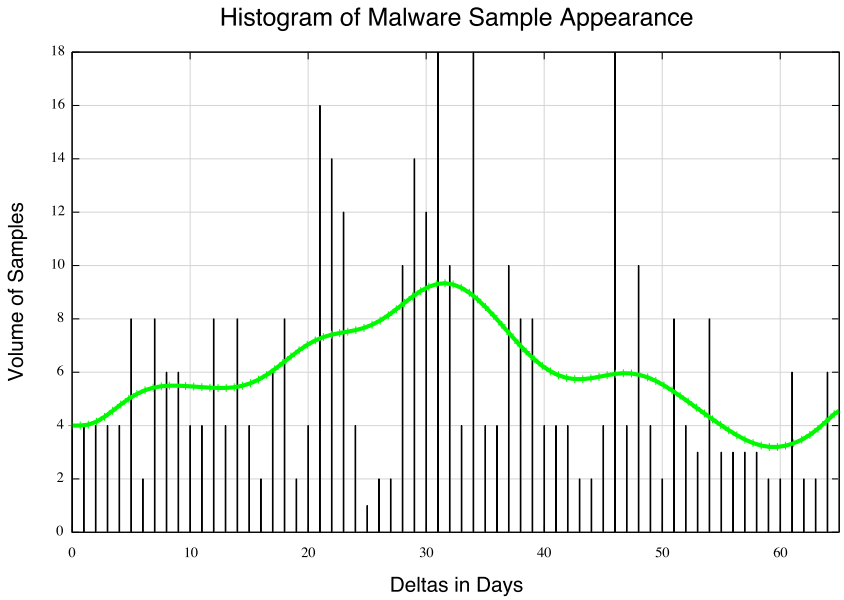


Figure 33: Volume of malware samples related with the DDoS botnet as they appeared in our feeds. Day zero is *05-20-2010*.

the middle of July we saw the biggest volume of malware samples related to the botnet's domain names surfacing in the security community. Finally, we should also note that these 18 domain names appeared in public blacklists after the take-down of the botnet was publicly disclosed (September 2010). Obviously, this was not exactly how we detected the botnet. After the initial identification of the 7 domain names in the beginning of May and with some very basic forensic analysis, we managed to quickly discover the entire corpus of the related domains.

In an effort to place Kopis' early detection abilities in comparison with recursive-based reputation systems (like Notos and Exposure) we check in the passive DNS database at ISC when these 18 domain names first appeared. Fifteen of them never showed up in the RDNSs that supply ISC with DNS data. The remaining three domain appeared for the first time on the following dates: **2010-06-24 06:56:34**, **2010-07-01 14:06:47** and **2010-09-08 04:32:36**. This means that the first domain

name related with this botnet appeared three months after the botnet was created and this would have been the earliest possible time that either Notos or Exposure could have detected these domain names assuming they were operating on passive DNS data from ISC — one of biggest passive DNS repositories worldwide. This clearly shows the need of detection systems like Kopis that can operate higher in the DNS hierarchy and provide Internet with an early global warning system for DNS.

Table 8: Number of the botnet related domain names that each feature family would have detected up-until the specified date assuming that the system was operating unsupervised.

Features/Dates	5/20	6/1	7/15	8/31
`All`	7	9	15	18
`RD+RP`	3	5	12	16
`IPR`	3	5	13	17
`RD+RP+AS(IPR)`	3	5	12	16

4.5 Summary

In recent years Internet miscreants have been leveraging the DNS to build malicious network infrastructures for malware command and control. In this Chapter we proposed a novel detection system called Kopis for detecting malware-related domain names. Kopis passively monitors DNS traffic at the upper levels of the DNS hierarchy, and is able to accurately detect malware-related domains by analyzing *global* DNS query resolution patterns.

Compared to previous DNS reputation systems such as Notos [7] and Exposure [11], which rely on monitoring traffic from *local* recursive DNS servers, Kopis offers a new vantage point and introduces new traffic features specifically chosen to leverage the *global* visibility obtained by monitoring network traffic at the upper DNS hierarchy. Unlike previous work Kopis enables DNS operators to *independently* (i.e., without the need of data from other networks) detect malware domains within their

authority, so that action can be taken to stop the abuse.

As a result, whereas previous systems can detect a malware-related domain only when there is an infected machine within the monitored network, Kopis can detect any domain within the monitored DNS zone. Kopis can thus enable network operators to take preemptive actions (e.g., blocking) even before any machine within their network is infected. Moreover, unlike previous work, Kopis can detect malware domains even when *no* IP reputation information is available.

We developed a proof-of-concept version of Kopis, and experimented with eight months of real-world data. Our experimental results show that Kopis can achieve high detection rates (e.g., 98.4%) and low false positive rates (e.g., 0.3% or 0.5%). In addition Kopis is able to detect new malware domains days or even weeks before they appear in public blacklists and security forums, and allowed us to discover the rise of a previously unknown DDoS botnet based in China.

CHAPTER V

DETECTING THE RISE OF DGA-BASED BOTNETS AT THE RECURSIVE LEVELS OF THE DNS HIERARCHY

5.1 Motivation

Malicious software, or *malware*, is the basis of most modern cyber-crimes. Cyber-criminals leverage malware-compromised machines to, for example, send spam, steal private information, host phishing webpages, perform denial of service (DoS) attacks, etc. In particular, Botnets are groups of malware-compromised machines, or *bots*, that can be remotely controlled by an attacker (the *botmaster*) through a *command and control* (C&C) communication channel.

Botnets have become the main platform for cyber-criminals to send spam, steal private information, host phishing web-pages, etc. Over time, attackers have developed C&C channels with different network structures. Most botnets today rely on a centralized C&C server, whereby bots query a predefined C&C domain name that resolves to the IP address of the C&C server from which commands will be received. Such centralized C&C structures suffer from the *single point of failure* problem because if the C&C domain is identified and taken down, the botmaster loses control over the entire botnet.

To overcome this limitation, attackers have used P2P-based C&C structures in botnets such as Nugache [86], Storm [97], and more recently Waledac [98], Zeus [3], and Alureon (a.k.a. TDL4) [36]. While P2P botnets provide a more robust C&C structure that is difficult to detect and take down, they are typically harder to implement and maintain. In an effort to combine the simplicity of centralized C&Cs with the robustness of P2P-based structures, attackers have recently developed a number of

botnets that locate their C&C server through *automatically generated* pseudo-random domains names. In order to contact the botmaster, each bot periodically executes a *domain generation algorithm* (DGA) that, given a random seed (e.g., the current date), produces a list of *candidate* C&C domains. The bot then attempts to resolve these domain names by sending DNS queries until one of the domains resolves to the IP address of a C&C server. This strategy provides a remarkable level of *agility* because even if one or more C&C domain names or IP addresses are identified and taken down, the bots will eventually get the IP address of the relocated C&C server via DNS queries to the next set of automatically generated domains. Notable examples of DGA-based botnets (or DGA-bots, for short) are Bobax [83], Kraken [72], Sinowal (a.k.a. Torpig) [84], Srizbi [74], Conficker-A/B [65], Conficker-C [60] and Murofet [77]. A defender can attempt to reverse engineer the bot malware, particularly its DGA algorithm, to pre-compute current and future candidate C&C domains in order to detect, block, and even take down the botnet. However, reverse engineering is not always feasible because the bot malware can be updated very quickly (e.g., hourly) and obfuscated (e.g., encrypted, and only decrypted and executed by external triggers such as time).

With the research presented in this this Chapter, we propose a novel detection system, called Pleiades, to identify DGA-based bots within a monitored network without reverse engineering the bot malware. Pleiades is placed "below" the local recursive DNS (RDNS) server or at the edge of a network to monitor DNS query/response messages from/to the machines within the network. Specifically, Pleiades analyzes DNS queries for domain names that result in *Name Error* responses [55], also called `NXDOMAIN` responses, i.e., domain names for which no IP addresses (or other resource records) exist. We will refer to these domain names as NXDomains.

The focus on NXDomains is motivated by the fact that modern DGA-bots tend to query large sets of domain names among which relatively few successfully resolve to

the IP address of the C&C server. Therefore, to automatically identify DGA domain names, Pleiades searches for relatively large clusters of NXDomains that (i) have similar syntactic features, and (ii) are queried by multiple potentially compromised machines during a given epoch.

The intuition is that in a large network, like the ISP network where we ran our experiments, multiple hosts may be compromised with the same DGA-bots. Therefore, each of these compromised assets will generate several DNS queries resulting in NXDomains, and a subset of these NXDomains will likely be queried by more than one compromised machine. Pleiades is able to automatically identify and filter out "accidental", user-generated NXDomains due to typos or mis-configurations. When Pleiades finds a cluster of NXDomains, it applies statistical learning techniques to build a model of the DGA. This is used later to detect future compromised machines running the same DGA and to detect *active domain names* that "look similar" to NXDomains resulting from the DGA and therefore probably point to the botnet C&C server's address.

Pleiades has the advantage of being able to discover and model new DGAs without labor-intensive malware reverse-engineering. This allows our system to detect new DGA-bots before any sample of the related malware family is captured and analyzed. Unlike previous work on DNS traffic analysis for detecting malware-related [8] or malicious domains in general [7, 11], Pleiades leverages *throw-away traffic* (i.e., unsuccessful DNS resolutions) to (1) discover the rise of new DGA-based botnets, (2) accurately detect bot-compromised machines, and (3) identify and block the active C&C domains queried by the discovered DGA-bots. Pleiades achieves these goals by monitoring the DNS traffic in local networks, without the need for a large-scale deployment of DNS analysis tools required by prior work.

Furthermore, while botnet detection systems that focus on network flow analysis [38, 87, 104, 110] or require deep packet inspection [33, 39] may be capable of

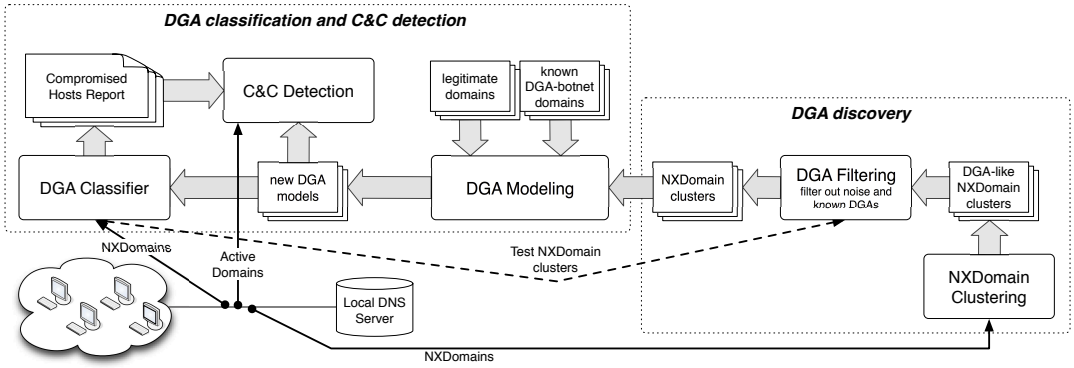


Figure 34: A high level overview of Pleiades.

detecting compromised machines within a local network, they do not scale well to the overwhelming volume of traffic typical in large ISP environments. On the other hand, Pleiades employs a *lightweight* DNS-based monitoring approach, and can detect DGA-based malware by focusing on a small fraction of all DNS traffic in an ISP network. This allows Pleiades to scale well to very large ISP networks, where we evaluated our prototype system.

5.1.1 Contributions

Pleiades is the first DGA-based botnet identification system that efficiently analyzes streams of unsuccessful domain name resolutions, or NXDomains, in large ISP networks to **automatically** identify DGA-bots, even in the absence of the corresponding malware. We built a prototype implementation of Pleiades, and evaluated its DGA identification accuracy over a large labeled dataset consisting of a mix of NXDomains generated by four different known DGA-based botnets and NXDomains "accidentally" generated by typos or mis-configurations. Our experiments demonstrate that Pleiades can accurately detect DGA-bots. Finally, we deployed and evaluated our Pleiades prototype in a large *production* ISP network for a period of 15 months. Our experiments discovered twelve new DGA-based botnets and enumerated the compromised machines. Half of these new DGAs have **never** been reported before.

5.2 A Detector for DGA-Based Botnets

In this section, we provide a high-level overview of our DGA-bot detection system Pleiades. As shown in Figure 34, Pleiades consists of two main modules: a *DGA Discovery* module, and a *DGA Classification and C&C Detection* module. We discuss the roles of these two main modules and their components, and how they are used in coordination to *actively learn* and update DGA-bot detection models. We describe these components in more detail in Sections 5.3 and 5.4.

5.2.1 DGA Discovery

The *DGA Discovery* module analyzes streams of unsuccessful DNS resolutions, as seen from "below" a local DNS server (see Figure 34). All NXDomains generated by network users are collected during a given epoch (e.g., one day). Then, the collected NXDomains are clustered according to the following two similarity criteria: (1) the domain name strings have similar statistical characteristics (e.g., similar length, similar level of "randomness", similar character frequency distribution, etc.) and (2) the domains have been queried by overlapping sets of hosts. The main objective of this NXDomain clustering process is to group together domain names that likely are automatically generated by the same algorithm running on multiple machines within the monitored network.

Naturally, because this clustering step is unsupervised, some of the output NXDomain clusters may contain groups of domains that happen to be similar by chance (e.g., NXDomains due to common typos or to mis-configured applications). Therefore, we apply a subsequent filtering step. We use a supervised *DGA Classifier* to prune NXDomain clusters that appear to be generated by DGAs that we have previously discovered and modeled, or that contain domain names that are similar to popular legitimate domains. The final output of the *DGA Discovery* module is a set of NXDomain clusters, each of which likely represents the NXDomains generated by

previously unknown or not yet modeled DGA-bots.

5.2.2 DGA Classification and C&C Detection

Every time a new DGA is discovered, we use a supervised learning approach to build models of what the domains generated by this new DGA "look like". In particular, we build two different statistical models: (1) a statistical multi-class classifier that focuses on assigning a specific DGA label (e.g., *DGA-Conficker.C*) to the *set of NXDomains* generated by a host h_i and (2) a Hidden Markov Model (HMM) that focuses on finding *single active domain names* queried by h_i that are likely generated by a DGA (e.g., *DGA-Conficker.C*) running on the host, and are therefore good *candidate C&C domains*.

The *DGA Modeling* component receives different sets of domains labeled as *Legitimate* (i.e., "non-DGA"), *DGA-Bobax*, *DGA-Torpig/Sinowal*, *DGA-Conficker.C*, *New-DGA-v1*, *New-DGA-v2*, etc., and performs the training of the multi-class *DGA Classifier* and the HMM-based *C&C Detection* module.

The *DGA Classification* module works as follows. Similar to the *DGA Discovery* module, we monitor the stream of NXDomains generated by each client machine "below" the local recursive DNS server.

Given a subset of NXDomains generated by a machine, we extract a number of statistical features related to the NXDomain strings. Then, we ask the *DGA Classifier* to identify whether this subset of NXDomains resembles the NXDomains generated by previously discovered DGAs. That is, the classifier will either label the subset of NXDomains as generated by a known DGA, or tell us that it does not fit any model. If the subset of NXDomains is assigned a specific DGA label (e.g., *DGA-Conficker.C*), the host that generated the NXDomains is deemed to be compromised by the related DGA-bot.

Once we obtain the list of machines that appear to be compromised with DGA-based bots, we take detection one step further. While all previous steps focused on NXDomains, we now turn our attention to domain names for which we observe valid resolutions. Our goal is to identify which domain names, among the ones generated by the discovered DGA-based bots, actually resolve into a valid IP address. In other words, we aim to identify the botnet's active C&C server.

To achieve this goal, we consider all domain names that are successfully resolved by a host which have been classified as running a given DGA, say *New-DGA-vX*, by the *DGA Classifier*. Then, we test these successfully resolved domains against the HMM specifically trained to recognize domains generated by *New-DGA-vX*. The HMM analyzes the sequence of characters that compose a domain name d, and computes the likelihood that d is generated by *New-DGA-vX*.

We use an HMM, rather than the *DGA Classifier*, because for the C&C detection phase we need to classify *single domain names*. The *DGA Classifier* is not suitable for this task because it expects as input *sets* of NXDomains generated by a given host to assign a label to the DGA-bot running on that host. Some of the features used by the *DGA Classifier* cannot be reliably extracted from a single domain name, which we discuss further in Sections 5.3.1.1 and 5.4.2.

5.3 DGA Discovery

The *DGA Discovery* module analyzes sequences of NXDomains generated by hosts in a monitored network, and in a *completely unsupervised* way, clusters NXDomains that are being automatically generated by a DGA. We achieve this goal in multiple steps (see Figure 34). First (*Step 1*), we collect sequences of NXDomains generated by each host during an epoch E. Afterwards (*Step 2*), we split the overall set of NXDomains generated by all monitored hosts into small subsets, and translate each set into a statistical feature vector (see Section 5.3.1.1 for details). We then apply the

X-means clustering algorithm [61] to group these domain subsets into larger clusters of domain names that have similar *string-based* characteristics.

Separately (*Step 3*), we cluster the NXDomains based on a completely different approach that takes into account whether two NXDomains are being queried by overlapping sets of hosts. First, we build a bipartite *host association* graph in which the two sets of vertices represent distinct hosts and distinct NXDomains, respectively. A host vertex V_{h_i} is connected to an NXDomain vertex V_{n_j} if host h_i queried NXDomain n_j. This allows us to identify different NXDomains that have been queried by overlapping sets of hosts. Intuitively, if two NXDomains are queried by multiple common hosts, this indicates that the querying hosts may be running the same DGA. We can then leverage this definition of similarity between NXDomains to cluster them (see Section 5.3.1.3 for details).

These two *distinct views* of similarities among NXDomains are then reconciled in a *cluster correlation* phase (*Step 4*). This step improves the quality of the final NXDomains clusters by combining the clustering results obtained in *Step 2* and *Step 3*, and to eliminate possible noise introduced by clusters of domains that may appear similar purely by chance, for example, similar typos originating from different network users.

The final clusters represent different groups of NXDomains, each containing domain names that are highly likely to be generated by the same DGA. For each of the obtained NXDomain clusters, the question remains if they belong to a known DGA, or a newly discovered one. To answer this question (*Step 5*), we use the *DGA Classifier* described in Section 5.4.2, which is specifically trained to distinguish between sets of NXDomains generated by currently known DGAs. Clusters that match previously modeled DGAs are discarded. On the other hand, if a cluster of NXDomains does not resemble any previously seen DGAs, we identify the cluster of NXDomains as having been generated by a new, previously unknown DGA. These NXDomains will

then be sent (*Step 6*) to the *DGA Modeling* module, which will update (i.e., re-train) the *DGA Classifier* component.

5.3.1 NXDomain Clustering

We now describe the *NXDomain Clustering* module in detail. First, we introduce the statistical features Pleiades uses to translate small sets of NXDomains into feature vectors, and then discuss how these feature vectors are clustered to find similar NXDomains.

5.3.1.1 Statistical Features

To ease the presentation of how the statistical features are computed, we first introduce some notation that we will be using throughout this section.

Definitions and Notation A domain name d consists of a set of labels separated by dots, e.g., `www.example.com`. The rightmost label is called the *top-level* domain (TLD or $TLD(d)$), e.g., `com`. The *second-level* domain (2LD or $2LD(d)$) represents the two rightmost labels separated by a period, e.g., `example.com`. The *third-level* domain (3LD or $3LD(d)$) contains the three rightmost labels, e.g., `www.example.com`, and so on.

We will often refer to splitting a sequence $NX = \{d_1, d_2, ..., d_m\}$ of NXDomains into a number of subsequences (or subsets) of length α, $NX_k = \{d_r, d_{r+1}, ..., d_{r+\alpha-1}\}$, where $r = \alpha(k-1)+1$ and $k = 1, 2, ..., \lfloor \frac{m}{\alpha} \rfloor$. Subscript k indicates the k-th subsequence of length α in the sequence of m NXDomains NX. Each of the NX_k domain sequences can be translated into a feature vector, as described below.

n-gram Features Given a subsequence NX_k of α NXDomains, we measure the frequency distribution of n-grams across the domain name strings, with $n = 1, .., 4$. For example, for $n = 2$, we compute the frequency of each 2-gram. At this point, we can compute the median, average and standard deviation of the obtained distribution

of 2-gram frequency values, thus obtaining three features. We do this for each value of $n = 1, ..., 4$, producing 12 statistical features in total. By measuring the median, average and standard deviation, we are trying to capture the *shape* of the frequency distribution of the n-grams.

Entropy-based Features This group of features computes the entropy of the character distribution for separate domain levels. For example, we separately compute the character entropy for the 2LDs and 3LDs extracted from the domains in NX_k. To better understand how these features are measured, consider a set NX_k of α domains. We first extract the 2LD of each domain $d_i \in NX_k$, and for each domain we compute the entropy $H(2LD(d_i))$ of the characters of its 2LD. Then, we compute the average and standard deviation of the set of values $\{H(2LD(d_i))\}_{i=1...\alpha}$. We repeat this for 3LDs and for the overall domain name strings. We measure a total of six features, which capture the "level of randomness" in the domains. The intuition is that most DGAs produce random-looking domain name strings, and we want to account for this characteristic of the DGAs.

Structural Domain Features This group of features is used to summarize information about the structure of the NXDomains in NX_k, such as their length, the number of unique TLDs, and the number of domain levels. In total, we compute 14 features. Specifically, given NX_k, we compute the average, median, standard deviation, and variance of the length of the domain names (four features), and of the number of domain levels (four features). Also, we compute the number of distinct characters that appear in these NXDomains (one feature), the number of distinct TLDs, and the ratio between the number of domains under the `.com` TLD and the number of domains that use other TLDs (two features). The remaining features measure the average, median, and standard deviation of the occurrence frequency distribution for the different TLDs (three features).

5.3.1.2 Clustering using Statistical Features

To find clusters of similar NXDomains, we proceed as follows. Given the set NX of all NXDomains that we observed from all hosts in the monitored network, we split NX into subsets of size α, as mentioned in Section 5.3.1.1. Assuming m is the number of distinct NXDomains in NX, we split the set NX into $\lfloor \frac{m}{\alpha} \rfloor$ different subsets where $\alpha = 10$. We motivate this choice for α in Appendix A.1.

For each of the obtained subsets NX_k of NX, we compute the aforementioned 33 statistical features. After we have translated each NX_k into its corresponding feature vector, we apply the X-means clustering algorithm [61]. X-means will group the NX_k into X clusters, where X is automatically computed by an optimization process internal to X-means itself. At this point, given a cluster $C = \{NX_k\}_{k=1..l}$ of l NXDomain subsets, we simply take the union of the NX_k in C as an NXDomain cluster.

5.3.1.3 Clustering using Bipartite Graphs

Hosts that are compromised with the same DGA-based malware naturally tend to generate (with high probability) partially overlapping sets of NXDomains. On the other hand, other "non-DGA" NXDomains are unlikely to be queried by multiple hosts. For example, it is unlikely that multiple distinct users make identical typos in a given epoch. This motivates us to consider NXDomains that are queried by several common hosts as similar, and in turn use this similarity measure to cluster NXDomains that are likely generated by the same DGA.

To this end, we build a sparse association matrix M, where columns represent NXDomains and rows represent hosts that query more than two of the column NXDomains over the course of an epoch. We discard hosts that query only one NXDomain to reduce the dimensionality of the matrix, since they are extremely unlikely to be running a DGA given the low volume of NXDomains they produce. Let a matrix

INPUT : Sparse matrix $M \in \Re^{l \times k}$, in which the rows represent l hosts and the columns represent k NXDomains.

[1] : Normalize M: $\forall j = 1, .., k \quad \sum_{i=1}^{l} M_{i,j} = 1$

[2] : Compute the similarity matrix S from M: $S = M^T \cdot M$

[3] : Compute the first ϱ eigenvectors from S by eigen-decomposition.

Let $U \in \Re^{\varrho \times k}$ be the matrix containing k vectors $u_1, ..., u_k$ of size ϱ resulting from the eigen-decomposition of S

(a vector u_i is a reduced ϱ-dimensional representation of the i-th NXDomain).

[4] : Cluster the vectors (i.e., the NXDomains) $\{u_i\}_{i=1,..,k}$ using the X-means algorithm

OUTPUT: Clusters of NXDomains

Algorithm 1: Spectral clustering of NXDomains.

element $M_{i,j} = 0$, if host h_i did not query NXDomain n_j. Conversely, let $M_{i,j} = w_i$ if h_i did query n_j, where w_i is a weight.

All non-zero entries related to a host h_i are assigned the same weight $w_i \sim \frac{1}{k_i}$, where k_i is the number of NXDomains queried by host h_i. Clearly, M can be seen as a representation of a bipartite graph, in which a *host vertex* V_{h_i} is connected to an *NXDomains vertex* V_{n_j} with an edge of weight w_i if host h_i queried NXDomain n_j during the epoch under consideration. The intuition behind the particular method we use to compute the weights w_i is that we expect that the higher the number of unique NXDomains queried by a host h_i (i.e., the higher k_i) the less likely the host is "representative" of the NXDomains it queries. This is in a way analogous to the *inverse document frequency* used in the text mining domain [1, 26].

Once M is computed, we apply a graph partitioning strategy based on spectral clustering [57, 58], as summarized in Algorithm 1. As a first step, we compute the first ρ eigenvectors of M (we use $\rho = 15$ in our experiments), and then we map each NXDomain (each column of M) into a ρ-dimensional vector. In effect, this mapping greatly reduces the dimensionality of the NXDomain vectors from the total number of hosts (the number of rows in M) to ρ. We then used the obtained ρ-dimensional NXDomain representations and apply X-means to cluster the NXDomains based on

their "host associations". Namely, NXDomains are grouped together if they have been queried by a similar set of hosts.

5.3.1.4 Cluster Correlation

We now have two complementary views of how the NXDomains should be grouped based on two different definitions of similarity between domain names. Neither view is perfect, and the produced clusters may still contain noise. Correlating the two results helps filter the noise and output clusters of NXDomains that are more likely to be generated by a DGA. Cluster correlation is performed in the following way.

Let $\mathcal{A} = \{A_1, .., A_n\}$ be the set of NXDomain clusters obtained by using statistical features, as described in Section 5.3.1.2, and $\mathcal{B} = \{B_1, .., B_m\}$ be the set of NXDomain clusters derived from the bipartite graph partitioning approach discussed in Section 5.3.1.3. We compute the intersection between all possible pairs of clusters $I_{i,j} = A_i \cap B_j$, for $i = 1, .., n$ and $j = 1, .., m$. All correlated clusters $I_{i,j}$ that contain less than a predefined number λ of NXDomains (i.e., $|I_{i,j}| < \lambda$) are discarded, while the remaining correlated clusters are passed to the DGA filtering module described in Section 5.3.2. Clusters that are not sufficiently agreed upon by the two clustering approaches are not considered for further processing. We empirically set $\lambda = 40$ in preliminary experiments.

5.3.2 DGA Filtering

The DGA filtering module receives the NXDomain clusters from the clustering module. This filtering step compares the newly discovered NXDomain clusters to domains generated by known DGAs that we have already discovered and modeled. If the NXDomains in a correlated cluster $I_{i,j}$ are classified as being generated by a known DGA, we discard the cluster $I_{i,j}$. The reason is that the purpose of the *DGA Discovery* module is to find clusters of NXDomains that are generated (with high probability) by a new, never before seen DGA. At the same time, this filtering step is responsible for

determining if a cluster of NXDomains is too noisy, i.e., if it likely contains a mix of DGA and "non-DGA" domains.

To this end, we leverage the *DGA Classifier* described in detail in Section 5.4. At a high level, we can treat the *DGA Classifier* as a function that takes as input a set NX_k of NXDomains, and outputs a set of tuples $\{(l_t, s_t)\}_{t=1..c}$, where l_i is a label (e.g., *DGA-Conficker.C*), and s_i is a score that indicates how confident the classifier is on attributing label l_i to NX_k, and c is the number of different classes (and labels) that the *DGA Classifier* can recognize.

When the DGA filtering module receives a new correlated cluster of NXDomains $I_{i,j}$, it splits the cluster into subsets of α NXDomains, and then passes each of these subsets to the *DGA Classifier*. Assume $I_{i,j}$ is divided into n different subsets. From the *DGA Classifier*, we obtain as a result n sets of tuples:

$$\{\{(l_t, s_t)\}^{(1)}_{t=1..c}, \{(l_t, s_t)\}^{(2)}_{t=1..c}, ..., \{(l_t, s_t)\}^{(n)}_{t=1..c}\}.$$

First, we consider for each set of tuples $\{(l_t, s_t)\}^{(k)}_{t=1..c}$ with $k = 1, .., n$, the label $\hat{l}^{(k)}$ that was assigned the maximum score. We consider a cluster $I_{i,j}$ as too noisy if the related labels $\hat{l}^{(k)}$ are too diverse. Specifically, a cluster is too noisy when the majority label among the $\hat{l}^{(k)}, k = 1, ..n$ was assigned to less than $\theta_{maj} = 75\%$ of the n domain subsets. The clusters that do not pass the θ_{maj} "purity" threshold will be discarded. Furthermore, NXDomain clusters whose majority label is the *Legitimate* label will also be discarded.

For each remaining cluster, we perform an additional "purity" check. Let the majority label for a given cluster $I_{i,j}$ be l^*. Among the set $\{\{(l_t, s_t)\}^{(k)}_{t=1..c}\}_{k=1..n}$ we take all the scores s_t whose related $l_t = l^*$. That is, we take the confidence score assigned by the classifier to the domain subsets that have been labeled as l^*, and then we compute the average $\mu(s_t)$ and the variance $\sigma^2(s_t)$ of these scores (notice that the scores s_t are in $[0, 1]$). We discard clusters whose $\sigma^2(s_t)$ is greater than a

predefined threshold $\theta_\sigma = 0.001$, because we consider the domains in the cluster as not being *sufficiently similar* to the majority label class.

At this point, if $\mu(s_t) < \theta_\mu$, with $\theta_\mu = 0.98$, we deem the NXDomain cluster to be not similar enough to the majority label class, and instead we label it as "new DGA" and pass it to the *DGA Modeling* module. On the other hand, if $\mu(s_t) \geq \theta_\mu$, we confirm the majority label class (e.g., *DGA-Conficker.C*) and do not consider it further.

The particular choice for the values of the above mentioned thresholds are motivated in Section 5.6.2.

5.4 DGA Classification and C&C Detection

Once a new DGA is reported by the *DGA Discovery* module, we use a supervised learning approach to learn how to identify hosts that are infected with the related DGA-based malware by analyzing the set of NXDomains they generate. To identify compromised hosts, we collect the set of NXDomains NX_{h_i} generated by a host, h_i, and we ask the *DGA Classifier* whether NX_{h_i} likely "belongs" to a previously seen DGA or not. If the answer is yes, h_i is considered to be compromised and will be labeled with the name of the (suspected) DGA-bot that it is running.

In addition, we aim to build a classifier that can analyze the set of active domain names, say AD_{h_i}, resolved by a compromised host h_i and reduce it to a smaller subset $CC_{h_i} \subset AD_{h_i}$ of likely C&C domains generated by the DGA running on h_i. Finally, the set CC_{h_i} may be manually inspected to confirm the identification of C&C domain(s) and related IPs. In turn, the list of C&C IPs may be used to maintain an IP blacklist, which can be employed to block C&C communications and mitigate the effects of the malware infection.

We now describe the components of the DGA classification and C&C detection module in more detail.

5.4.1 DGA Modeling

As mentioned in Section 5.3.2, the NXDomain clusters that pass the *DGA Filtering* and do not fit any known DGA model are assigned a *New-DGA-vX* label, where X is a unique identifier. At this point, we build two different statical models representative of *New-DGA-vX*: (1) a statistical multi-class classifier that can assign a specific DGA label to the set of NXDomains generated by a host h_i and (2) a Hidden Markov Model (HMM) that can compute the probability that a single *active* domain queried by h_i was generated by the DGA running on the host, thus producing a list of *candidate C&C domains*.

The *DGA Modeling* module takes as input the following information: (1) a list of popular legitimate domain names extracted from the top 10,000 domains according to `alexa.com`; (2) the list of NXDomains generated by running known DGA-bots in a controlled environment (see Section 5.5); (3) the clusters of NXDomains received from the *DGA Discovery* module. Let NX be one such newly discovered cluster of NXDomains. Because in some cases NX may contain relatively few domains, we attempt to extend the set NX to a larger set NX' that can help build better statistical models for the new DGA. To this end, we identify all hosts that "contributed" to the NXDomains clustered in NX from our sparse association matrix M and we gather all the NXDomains they generated during an epoch. For example, for a given host h_i that generated some of the domains clustered in NX, we gather all the other NXDomains domains NX'_{h_i} generated by h_i. We then add the set $NX' = \bigcup_i NX'_{h_i}$ to the training dataset (marked with the appropriate new DGA label). The reader may at this point notice that the set NX'_{h_i} may contain not only NXDomains generated by a host h_i due to running a DGA, but it may also include NXDomains "accidentally' generated by h_i. Therefore, this may introduce some noisy instances into the training dataset. However, the number of "accidental" NXDomains is typically very small, compared to the number of NXDomains generated by a DGA. Therefore, we rely on

the generalization ability of the statistical learning algorithms we use to smooth away the effects of this potential source of noise. This approach works well in practice, as we will show in Section 5.6.

5.4.2 DGA Classifier

The *DGA Classifier* is based on a multi-class version of the Alternating Decision Trees (ADT) learning algorithm [32]. ADT leverages the high classification accuracy obtained by Boosting [50], while producing compact classification rules that can be more easily interpreted.

To detect hosts that are compromised with DGA-based malware, we monitor all NXDomains generated by each host in the monitored network and periodically send this information to the *DGA Classifier*. Given a set NX_{h_i} of NXDomains generated by host h_i, we split NX_{h_i} into subsets of length α, and from each of these subsets we extract a number of statistical features, as described in Section 5.3.1.1 If one of these subsets of NXDomains is labeled by the *DGA Classifier* as being generated by a given DGA, we mark host h_i as compromised and we add its IP address and the assigned DGA label to a malware detection report.

5.4.3 C&C Detection

The *C&C Detection* module is based on Hidden Markov Models (HMM) [71]. We use one distinct HMM per DGA. Given the set $NX_{\mathcal{D}}$ of domains generated by a DGA $\mathcal{D}$, we consider each domain $d \in NX_{\mathcal{D}}$ separately, and feed these domains to an HMM for training. The HMM sees the domain names simply as a sequence of characters, and the result of the training is a model $HMM_{\mathcal{D}}$ that given a new domain name s in input will output the likelihood that s was generated by $\mathcal{D}$.

We use *left-to-right* HMM as they are used in practice to decrease the complexity of the model, effectively mitigating problems related to under-fitting. The HMM's emission symbols are represented by the set of characters allowed in valid domain

names (i.e., alphabetic characters, digits, '_', '-', and '.'). We set the number of hidden states to be equal to the average length of the domain names in the training dataset.

During operation, the *C&C Detection* module receives active domain names queried by hosts that have been previously classified by the *DGA Classifier* as being compromised with a DGA-based malware. Let h_i be one such host, and $\mathcal{D}$ be the DGA running on h_i. The *C&C Detection* module will send every domain s resolved by h_i to $HMM_{\mathcal{D}}$, which will compute a likelihood score $f(s)$. If $f(s) > \theta_{\mathcal{D}}$, s is flagged as a good candidate C&C domain for DGA $\mathcal{D}$.

The threshold $\theta_{\mathcal{D}}$ can be learned during the training phase. First, we train the HMM with the set $NX_{\mathcal{D}}$. Then, we use a set L of legitimate "non-DGA" domains from Alexa. For each domain $l \in L$, we compute the likelihood $f(l)$ and set the threshold $\theta_{\mathcal{D}}$ so to obtain a maximum target false positive rate (e.g., max FPs=1%).

5.5 Data Collection

In this section we provide an overview of the amount of NXDomain traffic we observed during a period of fifteen consecutive months (our evaluation period), starting on November 1st, 2010 and ending on January 15th, 2012. Afterwards, we discuss how we collected the domain names used to train and test our *DGA Classifier* (see Section 5.4).

5.5.1 NXDomain Traffic

We evaluated Pleiades over a 15-month period against DNS traffic obtained by monitoring DNS messages to/from a set of recursive DNS resolvers operated by a large North American ISP. These servers were physically located in the US, and served (in average) over 2 million client hosts per day[1]. Our monitoring point was "below" the

[1]We estimated the number of hosts by computing the average number of distinct client IPs seen per day.

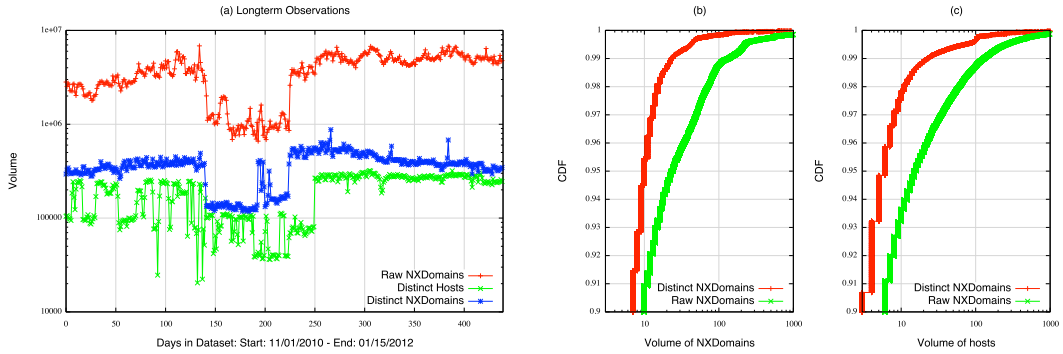


Figure 35: Observations from NXDomain traffic collected below a set of ISP recursive DNS servers over a 439 day window.

DNS servers, thus providing visibility on the NXDomains generated by the individual client hosts.

Figure 35(a) reports, per each day, (1) the number of NXDomains as seen in the *raw* DNS traffic, (2) the number of distinct hosts that in the considered day query at least one NXDomains, and (3) the number of distinct (de-duplicated) NXDomains (we also filter out domain names that do not have a valid effective TLD [42, 55, 56]). The abrupt drop in the number of NXDomains and hosts (roughly a 30% reduction) experienced between 2011-03-24 and 2011-06-17 was due to a configuration change at the ISP network.

On average, we observed about 5 millions "raw" NXDomains, 187,600 distinct hosts that queried at least one NXDomains, and 360,700 distinct NXDomains overall, per each day. Therefore, the average size of the association matrix M used to perform spectral clustering (see Section 5.3.1.3) was 187,600 $\times$ 360,700. However, it is worth noting that M is sparse and can be efficiently stored in memory. In fact, the vast majority (about 90%) of hosts query less than 10 NXDomains per day, and therefore most rows in M will contain only a few non-zero elements. This is shown in Figure 35(b), which reports the cumulative distribution function (CDF) for the volume of NXDomains queried by a host in the monitored network. On the other hand, Figure 35(c) shows the CDF for the number of hosts that query an NXDomain

Table 9: Detection results (in %) using 10-fold cross validation for different values of α.

	$\alpha = 5$ NXDomains			$\alpha = 10$ NXDomains		
Class	TP_{rate}	FP_{rate}	AUC	TP_{rate}	FP_{rate}	AUC
Bobax	95	0.4	97	**99**	**0**	**99**
Conficker	98	1.4	98	**99**	**0.1**	**99**
Sinowal	99	0.1	98	**100**	**0**	**100**
Murofet	98	0.7	98	**99**	**0.2**	**99**
Benign	96	0.7	97	**99**	**0.1**	**99**

(this relates directly to the sparseness of M according to its columns).

5.5.2 Ground Truth

In order to generate the ground truth to train and evaluate the *DGA Classifier* (Section 5.4), we used a simple approach. To collect the NXDomains generated by known DGA-based malware we used two different methods. First, because the DGA used by different variants of Conficker and by Murofet are known (derived through reverse-engineering), we simply used the respective algorithms to generate a set of domain names from each of these botnets. To obtain a sample set of domains generated by Bobax and Sinowal, whose exact DGA algorithm is not known (at least not to us), we simply executed two malware samples (one per botnet) in a VM-based malware analysis framework that only allows DNS traffic[2], while denying any other type of traffic. Overall we collected 30,000 domains generated by Conficker, 26,078 from Murofet, 1,283 from Bobax and, 1,783 from Sinowal.

Finally, we used the top 10,000 most popular domains according to `alexa.com`, with and without the `www.` prefix. Therefore, overall we used 20,000 domain names to represent the "negative" (i.e., "non-DGA") class during the training and testing of the *DGA Classifier*.

5.6 Evaluation

In this section, we present the experimental results of our system. We begin by demonstrating Pleiades' modeling accuracy with respect to known DGAs like Conficker, Sinowal, Bobax and Murofet. Then, we elaborate on the DGAs we discovered throughout the fifteen month NXDomain monitoring period. We conclude the section by summarizing the most interesting findings from the twelve DGAs we detected. Half of them use a DGA algorithm from a known malware family. The other half, to the best of our knowledge, have **no** known malware association.

5.6.1 DGA Classifier's Detection Results

In this section, we present the accuracy of the DGA classifier. We bootstrap the classifier with NXDomains from Bobax, Sinowal, Conficker-A, Conficker-B, Conficker-C and Murofet. We test the classifier in two modes. The first mode is bootstrapped with a "super" Conficker class composed of an equal number of samples from Conficker-A, Conficker-B and Conficker-C classes and another with each Conficker variant as its own class. As we mentioned in Section 5.4.2, the DGA classifier is based on a multi-class version of the Alternating Decision Trees (ADT) learning algorithm [32]. We build the vectors for each class by collecting NXDomains from one day of Honeypot traffic (in the case of Sinowal and Bobax) and one day of NXDomains produced by the DGAs for Conficker-A, Conficker-B, Conficker-C and Murofet. Finally, the domain names that were used to represent the benign class were the first 10,000 Alexa domain names with and without the `www.` child labels.

From the raw domain names in each of the classes, we randomly selected 3,000 sets of cardinality α. As a reminder, the values of α that we used were two, five, ten and 30. This was to build different training datasets in order to empirically decide which value of α would provide the best separation between the DGA models.

[2]We only allowed `UDP port 53`.

We generated additional testing datasets. The domain names we used in this case were from each class as in the case of the training dataset but we used different days. We do that so we get the minimum possible domain name overlap between the training and testing datasets. We evaluate the training datasets using two methods: 10-fold cross validation on the **training dataset** and by using the testing datasets computed from domains collected on **different days**. Both methods gave us very similar results. Our system performed the worst in the case of the 10-fold cross validation, therefore we chose to present this worst-case scenario.

In Table 9, we can see the detection results using two values for α, five and ten. We omit the results for the other values due to space limitations. The main confusion between the classes was observed in the datasets that contained separate Conficker classes, specifically between the classes of Conficker-A and Conficker-B. To address this problem, we created a generic Conficker class that had an equal number of vectors from each Conficker variant. This merging of the Conficker variants into a single "super" class allowed the DGA classifier to correctly classify 99.72% (Table 9) of the instances (7,986 correctly classified vs 22 incorrectly classified). Using the datasets with the five classes of DGAs, the weighted average of the TP_{rates} and FP_{rates} were 99.7% and 0.1%, respectively. As we see in Table 9, $\alpha = 5$ performs reasonably well, but with a higher rate of FPs.

5.6.2 NXDomain Clustering Results

In this section, we will discuss results from the DGA discovery module. In particular, we elaborate on the selection of the thresholds used, the unique clusters identified and the false alerts the DGA discovery module produced over the duration of our study.

5.6.2.1 Correlation Thresholds

In order to set the thresholds θ_{maj} and θ_{σ} defined in Section 5.3.2, we spent the first five days of November 2010 labeling the 213 produced clusters as DGA related

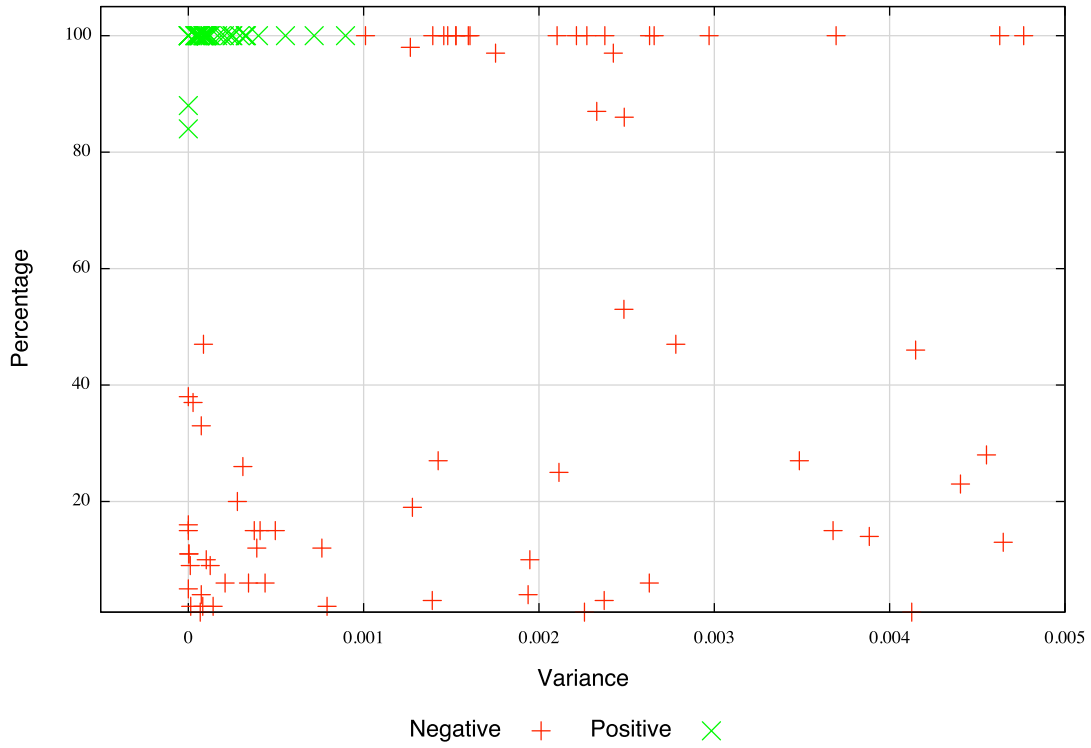


Figure 36: Thresholds θ_{maj} and θ_{σ} from the first five days of November 2010.

(Positive) or noisy (Negative). For this experiment, we included all produced clusters without filtering out those with θ_{μ}=98% (or higher) "similarity" to an already known one (see Section 5.3.2). In Figure 36, we can see in the Y-axis the percentage values for the dominant (non-benign) class in every cluster produced during these five days. In the X-axis we can see the variance that each dominant class had within each cluster. The results show that the Positive and Negative assignments had a clear cut, which we can achieve by setting the thresholds as $\theta_{maj} = 75\%$ and $\theta_{\sigma} = 0.001$. These thresholds gave us very good results throughout the duration of the experiments. As we will discuss in Section 5.6.2.3, the DGA discovery module falsely reported only five benign clusters over a period of 15 months. All falsely reported clusters had variance very close to 0.001.

5.6.2.2 New DGAs

Pleiades began clustering NXDomain traffic on the first day of November 2010. We bootstrapped the DGA modeler with domain names from already known DGAs but also a set of Alexa domain names as the benign class. In Table 10, we present all unique clusters we discovered throughout the evaluation period. The "Malware Family" column simply maps the variant to a known malware family if possible. We discover the malware family by checking the NXDomains that overlap with NXDomains we extracted from traffic obtained from a malware repository. Also, we manually inspected the clusters with the help of a security company's threat team. The "First Seen" column denotes the first time we saw traffic from each DGA variant. Finally, the "Population on Discovery" column shows the variant population on the discovery day. We can see that we can detect each DGA variant with an average number of 32 "infected hosts" across the entire statewide ISP network coverage.

Table 10: DGAs Detected by Pleiades.

Malware Family	First Seen	Population on Discovery
Shiz/Simda-C [80]	03/20/11	37
Bamital [35]	04/01/11	175
BankPatch [10]	04/01/11	28
Expiro.Z [30]	04/30/11	7
Bonnana [100]	08/03/11	24
Zeus.v3 [64]	09/15/11	39
New-DGA-v1	01/11/10	12
New-DGA-v2	01/18/11	10
New-DGA-v3	02/01/11	18
New-DGA-v4	03/05/11	22
New-DGA-v5	04/21/11	5
New-DGA-v6	11/20/11	10

As we see in Table 10, Pleiades reported six variants that belong to known DGA-enabled malware families [10, 30, 35, 64, 80, 100]. Six more variants of NXDomains were reported and modeled by Pleiades but for these, to the best of our knowledge, no known malware can be associated with them. A sample set of 10 domain names

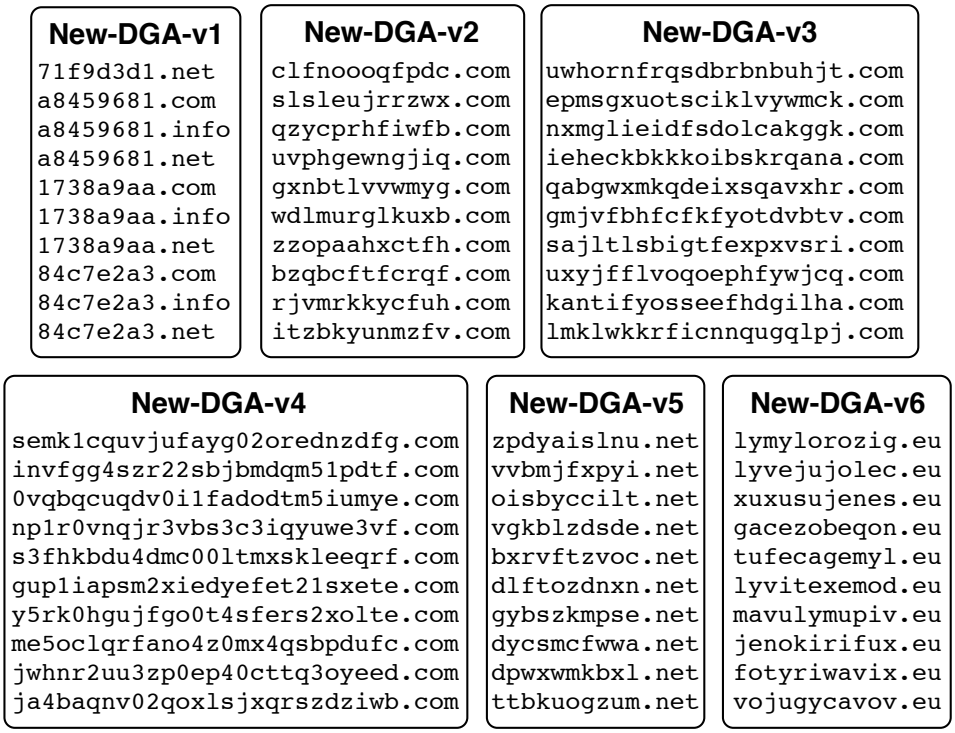


Figure 37: A sample of ten NXDomain for each DGA cluster that we could not associate with a known malware family.

for each one of these variants can be seen in Figure 37.

In the 15 months of our observations we observed an average population of 742 Conficker infected hosts in the ISP network. Murofet had the second largest population of infected hosts at 92 per day, while the Bonnana DGA comes third with an average population of 84 infected hosts per day. The fastest growing DGA is Zeus.v3 with an average population of 50 hosts per day, however, during the last four days of the experiments the Zeus.v3 DGA had an average number of 134 infected hosts. It is worth noting the New-DGA-v1 had an average of 19 hosts per day, the most populous of the newly identified DGAs.

5.6.2.3 False Reports on New DGAs

During our evaluation period we came across five categories of clusters falsely reported as new DGAs. In all of the cases, we modeled these classes in the DGA modeler as variants of the benign class. We now discuss each case in detail.

The first cluster of NXDomains falsely reported by Pleiades were random domain names generated by Chrome [49, 107]. Each time the Google Chrome browser starts, it will query three "random looking" domain names. These domain names are issued as a DNS check, so the browser can determine if NXDomain rewriting is enabled. The "Chrome DGA" was reported as a variant of Bobax from Pleiades. We trained a class for this DGA and flagged it as benign. One more case of testing for NXDomain rewriting was identified in a brand of wireless access points. Connectify[3], offers wireless hot-spot functionality and one of their configuration option enables the user to hijack the ISP's default NXDomain rewriting service. The device generates a fixed number of NXDomains to test for rewriting.

Two additional cases of false reports were triggered by domain names from the `.it` and `.edu` TLDs. These domain names contained minor variations on common words (i.e. repubblica, gazzetta, computer, etc.). Domain names that matched these clusters appeared only for two days in our traces and never again. The very short lived presence of these two clusters could be explained if the domain names were part of a spam-campaign that was remediated by authorities before it became live.

The fifth case of false report originated from domain names under a US government zone and contained the string `wpdhsmp`. Our best guess is that these are internal domain names that were accidentally leaked to the recursive DNS server of our ISP. Domain names from this cluster appeared only for one day. This class of NXDomains was also modeled as a benign variant. It is worth noting that all falsely reported DGA

[3] `www.connectify.me`

Table 11: TPs (%) for C&C detection (1,000 training sequences).

	FPs (%)					
botnet	*0.1*	*0.5*	*1*	*3*	*5*	*10*
Zeus.v3	99.9	99.9	99.9	99.9	99.9	99.9
Expiro.Z	33.03	64.56	78.23	91.77	95.23	98.67
Bamital	100	100	100	100	100	100
Shiz	0	1.64	21.02	96.58	100	100
Bonnana	3.8	10.69	15.59	27.67	35.05	48.43
BankPatch	56.21	70.77	93.18	99.9	99.91	99.94

clusters, excluding the Chrome cluster, were short lived. If operators are willing to wait a few days until a new DGA cluster is reported by Pleiades, these false alarms would not have been raised.

5.6.3 C&C Detection

To evaluate the effectiveness of the *C&C Detection*, we proceeded as follows. We considered the six new DGAs which we were able to attribute to specific malware, as shown in Table 11. Let NX_i be the set of NXDomains collected by the *DGA Discovery* (Section 5.3) and *DGA Modeling* (Section 5.4.1) modules for the i-th DGA. For each DGA, we set aside a subset $NX_i^{train} \subset NX_i$ of NXDomains to train an HMM_i model. Then we use the remaining $NX_i^{test} = NX_i - NX_i^{train}$ to compute the true positive (TP) rate of HMM_i, and a set A that consists of 602,969 unique domain names related to the consistently popular domain names according to `alexa.com` to compute the false positive (FP) rate. To obtain A we first consider all domain names that have been consistently ranked in the top 100,000 popular domains by `alexa.com` for approximately one year. This gave us a set T of about 60,000 "stable" popular domain names, which we consider as *legitimate* domains. Then, we monitored the stream of successful DNS queries in a large live network for a few hours, and we added to A all the domain names whose effective 2LD is in T.

We performed experiments with a varying number $c = |NX_i^{train}|$ of training samples. Specifically, we set c equal to 100, 200, 500, 1,000, 2,000, 5,000, and 10,000. We then computed the trade-off between TPs and FPs for different detection thresholds.

In the interest of space, we report only the results for c=1,000 in Table 11. In general, the results improve for increasing numbers of training instances. We set the detection threshold so as to obtain an FP rate equal to 0.1%, 0.5%, 1%, 3%, 5%, and 10%. As we can see, at FP=1% we obtained a high (> 93%) TP rate for three out of six DGAs, and relatively good results (> 78%) in five out of six cases. At FP=3% we have high TP rate (> 91%) in five out of six cases.

As mentioned in Section 5.2, the *C&C Detection* module reduces the set of domain names successfully resolved by a host h that have been labeled as compromised with DGA-malware to a smaller set of good *candidate C&C domains* generated by the DGA. The results in Table 11 show that if we rank the domains resolved by h according to the likelihood assigned by the HMM, in most cases we will only need to inspect between 1/100 to 3/100 of the active domains queried by h to discover the C&C.

5.6.4 Case Studies

5.6.4.1 Zeus.v3

In September 2011, Pleiades detected a new DGA that we linked to the Zeus.v3 variant a few weeks later. The domain names collected from the machines compromised by this DGA-malware are hosted in six different TLDs: `.biz`,`.com`,`.info` ,`.net` ,`.org` and `.ru`. Excluding the top level domains, the length of the domain names generated by this DGA are between 33 and 45 alphanumeric characters. By analyzing one sample of the malware[4] we observed that its primary C&C infrastructure is P2P-based. If the malware fails to reach its P2P C&C network, it follows a contingency plan, where a DGA-based component is used to try to recover from the loss of C&C communication. The malware will then resolve pseudo-random domain names, until an active C&C domain name is found.

[4]Sample MD5s: 8f60afa9ea1e761edd49dfe012c22cbf and ccec69613c71d66f98abe9cc7e2e20ef.

To date, we have discovered 12 such C&C domains. Over time, these 12 domains resolved to five different C&C IPs hosted in four different networks, three in the US (AS6245, AS16626 and AS3595) and one in the United Kingdom (AS24931). Interestingly, we observed that the UK-based C&C IP address remained active for a very short period of time of only a few minutes, from Jan 25, 2012 `12:14:04` EST to Jan 25, 2012 `12:22:37` EST. The C&C moved from a US IP (AS16626) to the UK (AS24931), and then almost immediately back to the US (AS3595).

5.6.4.2 BankPatch

We picked the BankPatch DGA cluster as a sample case for analysis since this botnet had been active for several months during our experiments and the infected population continues to be significant. The C&C infrastructure that supports this botnet is impressive. Twenty six different clusters of servers acted as the C&Cs for this botnet. The botnet operators not only made use of a DGA and also moved the active C&Cs to different networks every few weeks (on average). During our C&C discovery process, we observed IP addresses controlled by a European CERT. This CERT has been taking over domain names from this botnet for several months. We managed to cross-validate with them the completeness and correctness of the C&C infrastructure. The complete information about the C&C infrastructure can be found in Table 12.

The actual structure of the domain name used by this DGA can be separated into a four byte prefix and a suffix string argument. The suffix string arguments we observed were:

`seapollo.com`, `tomvader.com`, `aulmala.com`, `apontis.com`, `fnomosk.com`, `erhogeld.com`, `erobots.com`, `ndsontex.com`, `rtehedel.com`, `nconnect.com`, `edsafe.com`, `berhogeld.com`, `musallied.com`, `newnacion.com`, `susaname.com`, `tvolveras.com` and `dminmont.com`.

The four bytes of entropy for the DGA were provided by the prefix. We observe

collisions between NXDomains from different days, especially when only one suffix argument was active. Therefore, we registered a small sample of ten domain names at the beginning of 2012 in an effort to obtain a glimpse of the overall distribution of this botnet. Over a period of one month of monitoring the sink-holed data from the domain name of this DGA, this botnet has infected hosts in 270 different networks distributed across 25 different countries. By observing the recursive DNS servers from the domain names we sinkholed, we determined 4,295 were located in the US. The recursives we monitored were part of this list and we were able to measure 86 infected hosts (on average) in the network we were monitoring. The five countries that had the most DNS resolution requests for the sinkholed domain names (besides the US) were Japan, Canada, the United Kingdom and Singapore. The average number of recursive DNS servers from these countries that contacted our authorities was 22 — significantly smaller that the volume of recursive DNS servers within the US.

5.7 Summary

Given a known botnet command and control (C&C) domain, botnet traffic can be trivially detected and blocked. Indeed, many botnet detection systems employ a blacklist of known command and control (C&C) domains to detect bots and block their traffic. Similar to signature-based virus detection, such botnet detection approach is static because the blacklist is updated only after an external (and often manual) process of domain discovery.

As a response, botmasters have begun employing domain generation algorithms (DGAs) to dynamically produce a large number of random domain names and select a small subset for actual C&C use. That is, a C&C domain is randomly generated and used for a very short period of time, thus rendering detection approaches that rely on static domain lists ineffective. Naturally, if we know how a domain generation algorithm works, we can generate the domains ahead of time and still identify

Table 12: C&C Infrastructure for BankPatch.

IP addresses	CC	Owner
146.185.250.{89-92}	RU	Petersburg Int.
31.11.43.{25-26}	RO	SC EQUILIBRIUM
31.11.43.{191-194}	RO	SC EQUILIBRIUM
46.16.240.{11-15}	UA	iNet Colocation
62.122.73.{11-14,18}	UA	"Leksim" Ltd.
87.229.126.{11-16}	HU	Webenlet Kft.
94.63.240.{11-14}	RO	Com Frecatei
94.199.51.{25-18}	HU	NET23-AS 23VNET
94.61.247.{188-193}	RO	Vatra Luminoasa
88.80.13.{111-116}	SE	PRQ-AS PeRiQuito
109.163.226.{3-5}	RO	VOXILITY-AS
94.63.149.{105-106}	RO	SC CORAL IT
94.63.149.{171-175}	RO	SC CORAL IT
176.53.17.{211-212}	TR	Radore Hosting
176.53.17.{51-56}	TR	Radore Hosting
31.210.125.{5-8}	TR	Radore Hosting
31.131.4.{117-123}	UA	LEVEL7-AS IM
91.228.111.{26-29}	UA	LEVEL7-AS IM
94.177.51.{24-25}	UA	LEVEL7-AS IM
95.64.55.{15-16}	RO	NETSERV-AS
95.64.61.{51-54}	RO	NETSERV-AS
194.11.16.133	RU	PIN-AS Petersburg
46.161.10.{34-37}	RU	PIN-AS Petersburg
46.161.29.102	RU	PIN-AS Petersburg
95.215.{0-1}.29	RU	PIN-AS Petersburg
95.215.0.{91-94}	RU	PIN-AS Petersburg
124.109.3.{3-6}	TH	SERVENET-AS-TH-AP
213.163.91.{43-46}	NL	INTERACTIVE3D-AS
200.63.41.{25-28}	PA	Panamaserver.com

and block botnet C&C traffic. The existing solutions are largely based on reverse engineering of the bot malware executables, which is not always feasible.

The research presented in this Chapter introduced a new technique to detect randomly generated domains without reversing malware. Our insight is that most of the DGA-generated (random) domains that a bot queries would result in Non-Existent Domain (NXDomain) responses, and that bots from the same botnet (with the same DGA algorithm) would generate similar NXDomain traffic.

Our approach uses a combination of clustering and classification algorithms. The clustering algorithm clusters domains based on the similarity in the make-ups of domain names as well as the groups of machines that queried these domains. The

classification algorithm is used to assign the generated clusters to models of known DGAs. If a cluster cannot be assigned to a known model, then a new model is produced, indicating a new DGA variant or family.

We implemented a prototype system and evaluated it on real-world DNS traffic obtained from large ISPs in North America. We reported the discovery of twelve DGAs. Half of them are variants of known (botnet) DGAs, and the other half are brand new DGAs that have never been reported before.

CHAPTER VI

CONCLUSION

6.1 Overall Contributions and Summary

The goal of this thesis has been the delivery of methods and algorithms that could make an early warning system for DNS-based illicit operations feasible. To achieve this goal we had to address the following three research problems:

- *Dynamically* assign reputation scores to domain names according to their successful DNS resolutions.
- *Dynamically* detect new malware-related domains in various parts of the DNS hierarchy.
- *Dynamically* identify the rise of "Domain name Generation Algorithm"-based (or DGA) botnets, using properties of the DGA-based botnets communication cycle observable at the local recursive DNS level.

To address the limitation of static domain name black lists we introduced Notos [7], a dynamic reputation system for DNS. The system uses passive DNS evidence from recursive DNS servers to statistically identify benign and malicious domain names. Notos can statistically correlate the two planes in DNS: the name space and the address space. The primary goal of Notos is to automatically assign a low reputation score to a domain that is involved in malicious activities, such as malware spreading, "phishing", and spam campaigns. Conversely, we want to assign a high reputation score to domains that are used for legitimate purposes.

Given the nature of emerging Internet threats, it is important that we develop global monitoring capabilities to stop impending attacks on our networks. To address

the need of an early warning system across the DNS hierarchy we split this task into two steps. Firstly, the timely detection of malware-related domain names at the higher levels of the DNS hierarchy. Secondly, the detection of DGA-enabled malware domain names at the recursive level.

To that extent we proposed Kopis [8], which operates in the upper layers of the DNS hierarchy and is capable of detecting malware-related domain names "on-the rise". Kopis is the first component of our early warning system, which can be operated independently by the TLD and ANS operators and report on malicious domain names. Kopis enables DNS operators to *independently* deploy the system and detect malware-related domains from within their authority zones without the need for external data from other networks or other inter-organizational coordination.

The second component of our early warning system enables the detection of new emerging DGA-based botnets. Such detection systems should be able to operate at the DNS recursive level or at the edge of a network, so operators can independently identify DGA-based compromised machines. To that extent we proposed Pleiades [9], a detection system that is capable of reporting the rise of DGA botnets in a local network. This is made possible by statistical modeling of the unsuccessful DNS resolutions at the recursive DNS level of the monitored network. Additionally, Pleiades, is also able to learn models from traffic generated from already known DGA-based malware and detect active DGA-based infections in the local network.

In summary, this thesis is able to provide answers to all three research problems noted in the beginning of this Section. In detail, with Notos (Chapter 3) we proposed methods to quantify and track dynamically changing reputations for DNS [7] based on passive network measurements. In order to address the need of early warning systems for DNS, we proposed two novel detection systems, Kopis [8] and Pleiades [9] — in Chapter 4 and 5 respectively. These two early warning systems for DNS enable the security community to identify emerging threats (e.g., new botnets), across the DNS

hierarchy (i.e., at the recursive and at the TLD/ANS levels) in a timelier manner.

6.2 Considerations and Limitations

In this section we discuss key considerations for the detection systems presented in this thesis.

6.2.1 Limitation of Notos

This section discusses the limits of Notos, and the potential for evasion in real networks.

6.2.1.1 Passive DNS Data Collection and IPv6 Considerations

One of the main limitations is the fact that Notos is unable to assign reputation scores to domain names in resource records that have no historic (passive DNS) information at the host and BGP prefix levels. We should also note that sufficient time (i.e., for an ISP a few weeks) is necessary for data collection and a diverse passive DNS monitoring capability is also mandatory to create a passive DNS data collection repository.

This passive DNS repository is critical for Notos, since the statistical vectors are computed based on resource records associations within the repository. Therefore, if an attacker purchases new domain names and new address space (in new BGP prefixes), and never reuses old domain names or old IP addresses for his illicit operations, Notos will not be able to accurately assign a low reputation score to the new domains.

However, in the IPv4 address space, such a scenario is very unlikely to happen due to the impending exhaustion of the available address space. This claim will not hold when we move for IPv4 to IPv6. Such a transition will cause problems for the reputation assignment process for Notos. This will be a direct artifact of the statistical features we extract with the proposed systems, which are based on IP granularity. However, we believe that the features based on BGP prefixes and AS numbers, which we also introduce in our system, would still be able to capture some of the attacker's

DNS agility, which is typical now and should be also part of the malicious DNS hosting behavior in the IPv6 era. These BGP and AS based statistical features should be considered as the bases of new feature sets that would be able to statistically describe the DNS abuse in IPv6.

Summarizing, as long as newly generated domain names share some network properties (e.g., IPs or BGP prefixes) with already labeled RRs, Notos will be able to assign an accurate reputation score. In particular, since network resources are finite and more expensive to renew or change, even if the domain properties change, Notos can still identify whether a domain name may be associated with malicious behavior. In addition, if a given domain name for which we want to know the reputation is not present in the passive DNS DB, we can actively probe it, thus forcing a related passive DNS entry. However, this is possible only when the domain successfully maps to a non-empty set of IP addresses.

6.2.1.2 DNS Spoofing/Poisoning Considerations

The passive DNS data collection methodology that feed data to the passive DNS database can potentially be an evasion vector for our system. The positioning of the DNS sensor and avoidance of DNS poisoning is very important to maintain consistent passive DNS information for all zones in the database. More specifically, if the passive DNS sensor is placed above the recursive DNS server, it is prone to poisoning attacks especially if the sensor does not validate each DNS transaction for correctness. On the other hand, placing the passive DNS sensor above the recursive will add extra merit to the passive DNS information that can be collected since all authority related information (Name Server information) blended within the authority section of the DNS resolution, which then can be easily mined. This is not the case if the passive DNS sensor is placed below that recursive DNS server. It is up to the configuration of the recursive DNS software (e.g., BIND) to include authoritative information in

the DNS resolution returned to the stub resolver (or initiator of the DNS query). Monitoring the DNS traffic below the recursive makes DNS poisoning significantly harder due to ingress filtering [28, 29] in the routers between the recursive and the internal attacker who tries to poison the passive DNS monitoring.

6.2.1.3 False Positives Considerations

Our experimental results using the top 10,000 Alexa domain names as *known good* domains, report a false positive fate of 0.4%. While low in percentage, the absolute number of false positives may become significant in those cases in which very large numbers of new domain names are fed to Notos on a daily basis (e.g., in case of deployment in a large ISP network). However, we envision our Notos reputation system to be used not as a stand-alone system, but rather in cooperation with other defense mechanisms. For example, Notos may be used in collaboration with spam-filtering systems [73], systems that classify URL/URI information [62] or systems that examine binary downloads [79] and other type of Internet abuse that also use DNS (e.g., search poisoning [53] or malicious ad-networks [76]).

For example, if an email contains a link to a website whose domain name has a low reputation score according to Notos, the spam filter can increase the total spam-score of the email. However, if the rest of the email appears to be benign, the spam filter may still decide to accept the email. The same analogy can be made when someone examines unknown URLs/URIs. In order to further reduce FPs from such system, someone may use reputation score on the domain names part of the URL.

During manual analysis of false positives encountered in our evaluations and during operational deployment in Damballa Inc., we were able to draw some interesting observations. We found that a number of legitimate sites (e.g., goldsgym.com, blackhat.com) are being hosted in networks that historically have been linked with significant number of malicious domain names. In these cases, Notos, will tend to penalize

the reputation of such legitimate domains because they simply reside in a historically *bad Internet neighborhood*. In time, the reputation score assigned to these domains may change, if the administrators of the network in which the benign domain names are hosted take actions to "clean up" their networks.

Sinkholes and Parking sites: The second category of false positives that have been seen during the operational deployment of Notos is related to parking sites and sinkholes. Briefly, a parking site is an IP address that is used to "park" domain names after they are expired or taken-over by the registrars. Unfortunately, in the case of malware-related domain names, once a malicious domain name is taken-over, the malware will still try to "phone-home" to what the malware still considers as the active C&C domain. Clearly, from the passive evidence point of view, the parking Internet host will now be directly associated with a large volume of malware. This will cause its reputation to decrease and penalize the remaining domain names pointing to the same parking host. Fortunately, such parking hosts are trivially detectable, due to the abnormally high volume of domain names that point into them. Therefore, this hand-full of hosts could simply be excluded from the RHIP list during the vector computation process.

In the case of sinkholes, a similar problem arises. Briefly, a sinkhole is an Internet host in which malicious domain names are delegated after a domain take over action is performed. Typically, sinkholes are hosted in networks operated by "white-hats". As in the case of parking sites, sinkholes will attract an abnormally high volume of malware reaching out to them. In some cases this might degrade the reputation of the entire network. For example, if the sinkhole resides in an academic network (e.g., Georgia Institute of Technology), the probability of coming across more malware reaching out to hosts within the same BGP prefix or the AS that Georgia Tech operates is very small. Therefore, when we compute a reputation vector of other domain

names in the same BGP prefix or AS, the sinkhole infrastructure will not negatively bias the reputation vector simply because the standard deviation and variance on the evidence vectors both on the host, BGP and AS levels would not resemble a typical evidence vector from a low reputation network. However, if the sinkhole resides in a"white-hat" hosting provider that happens to occasionally experience sporadic cases of Internet abuse, in that case the sinkhole infrastructure will negatively impact the reputation of the entire BGP prefix and perhaps that AS as well in which the hosting provider resides in. Maybe the easiest way around this problem is for the few sinkhole operators to clearly separate their infrastructure from the rest of the "white-hats" hosting infrastructure.

6.2.2 Limitation of Kopis

In this section, we elaborate on possible evasion techniques and discuss some operational issues of Kopis.

6.2.2.1 Evasion techniques

Kopis relies significantly on the *Requester Diversity* (RD) and *Requester Profile* features. An attacker may attempt to dilute the information provided by the RP and RD features to evade Kopis. This could be achieved by resolving domain names from a diverse set of *open recursive* DNS servers or even from random IPs acting as stub resolver (e.g., using infected machines). This will not be as easy as it sounds, due to the RP feature family. This is because even if the adversary looks up domain names from various different IP addresses, the adversary will still have to look up a large number of domain names under the same authority to make the weight of each requester large enough to alter the RP features. Additionally, the adversary will have to repeatedly (for a long enough period of time) ask for different domain names served by the same authority in order to influence/dilute the RDNS weighing function.

In order to be able to artificially create the necessary signal that may dilute

or even disturb the modeling of legitimate and malware-related domain names, the adversary would have to obtain access to traffic at the authority name or TLD servers. Furthermore, the adversary would need a full list of statistical feature values used from Kopis. Such an attack would be similar in spirit to polymorphic blending attacks [31]. We note here that reliable and systematic access to DNS traffic at the authoritative or TLD level is extremely hard to obtain, since it would require the collaboration of the registrar that controls the AuthNS or the TLD servers.

Domain name generation algorithms (DGAs) have been used by malware families (i.e., Conficker [65], Zeus/Murofet [77], Bobax [83], Torpig [84] etc.) in the last few years. The new seed of these DGAs typically has the periodicity of a day. This implies that domain names generated by DGAs (and under the zones Kopis monitors) will be active only for a small period of time (e.g., a day). Due to the daily observation period mandatory for Kopis to provide detection results, such malware-related domain names will be potentially inactive by the time they are reported by our detection system. Perhaps, operating Kopis with smaller epochs (i.e., hourly granularity) could potentially solve this problem, however we leave the validation of this claim as part of our future work.

6.2.2.2 TLDs and Domain Registrars

As we have already discussed, just observing the DNS resolution requests at the TLD level will not provide sufficient information for the system to reconstruct the IP addresses mapped with the queried domain names. There are several ways to resolve this issue. The simplest way to reconstruct the IP addresses for a given domain name is to check a large passive DNS database. For the domains that are not replicated in the passive DNS database, we can use an *active probing* strategy to retrieve the resolved IP addresses with little overhead.

Several TLDs are anycasted. This implies that by just monitoring one of potential

several TLDs physical servers will not be sufficient to capture the full network and geographical diversity of the requesters. In cases where TLD servers are anycasted we need to assume that the TLD operator will have the ability to reconstruct the traces from the vast majority of those anycasted servers in order to formulate an accurate enough RP and RD statistical features for each domain name.

As a final classification heuristic, especially in the case of domain registrars, they can potentially combine Kopis with domain name registration information. Classification results from Kopis can be combined with domain name registration information (trivially accessible to domain registrars) in order to further reduce FPs but also provide an additional correlation between domain registration accounts that own domains with suspicious resolution behavior according to Kopis.

A different evasion approach may be that the adversary randomly looks up domain names that has nothing to do with the malware (i.e., C&C communication). In this case the adversary will manage to introduce some additional traffic towards these domain names. The final goal would be to evade the signal that corresponds to the real C&C communication domain(s). In reality such a goal is hard to be achieved for a couple of reasons. Firstly, even if the non-C&C domain names were under the same ANS, the adversary needs to mimic all RD and RP classification signal that corresponds to benign domain name resolutions. In practice, the adversary controls only the machines that were infected, so it's hard to decide a priori the exact levels of noise signal necessary, so the adversary can successfully blend the new C&C domain name resolutions into the artificially created noise.

Secondly, in order for the adversary to be able to artificially create the necessary signal that may dilute or even disturb the modeling of legitimate and malicious domain names, the adversary would have to obtain access to traffic at the authority name or TLD servers. Furthermore, the adversary would need a full list of statistical feature values used from Kopis.

6.2.3 Limitation of Pleiades

Pleiades has some limitations. For example, once a new DGA is discovered, Pleiades can build fairly accurate statistical models of how the domains generated by the DGA "look like", but it is unable to learn or reconstruct the exact domain generation algorithm. Therefore, Pleiades will generate a certain number of false positives and false negatives. However, the results we presented in Table 9 show that Pleiades is able to construct a very accurate *DGA Classifier* module, which produces very few false positives and false negatives for $\alpha = 10$. At the same time, Table 11 shows that the *C&C Detection* module, which attributes single active domain names to a given DGA, also works fairly well in a majority of cases. Unfortunately, there are some scenarios in which the HMM-based classification has difficulties. We believe this is because our HMM considers domain names simply to be sequences of individual characters. In our future work, we plan to experiment with 2-grams, whereby a domain name will be seen as a sequence of pairs of characters, which may achieve better classification accuracy for the harder to model DGAs.

As we mentioned in Section 5.2, detecting active DGA-generated C&C domains is valuable because their resolved IP addresses can be used to update a C&C IP blacklist. In turn, this IP blacklist can be used to block C&C communications at the network edge, thus providing a way to mitigate the botnet's malicious activities. Clearly, for this strategy to be successful, the frequency with which the C&C IP addresses change should be lower than the rate with which new pseudo-random C&C domain names are generated by the DGA. This assumption holds for all practical cases of DGA-based malware we encountered. After all, the generation of pseudo-random domains mainly serves the purpose of making the takedown of loosely centralized botnets harder. However, one could imagine "hybrid" botnets that use DGA-generated domains to identify a set of peer IPs to bootstrap into a P2P-based C&C infrastructure. Alternatively, the DGA-generated C&C domains may be flux domains, namely domain

names that point to an IP fluxing network. It is worth noting that such sophisticated "hybrid" botnets may be quite complex to develop, difficult to deploy, and hard to manage successfully.

Another potential limitation is due to the fact that Pleiades is not able to distinguish between different botnets whose bot-malware use the same DGA algorithm. In this case, while the two botnets may be controlled by different entities, Pleiades will attribute the compromised hosts within the monitored network to a single DGA-based botnet.

6.3 Open Research Problems

With the research presented in this thesis, researchers can now make use of a variety of classification signals in the context of DNS abuse. All three detection systems presented here make use of those classification signals to effectively improve the state of the art of classification systems that have to address DNS abuse. After conceptualizing, implementing and deploying these classifications system in an operational environment we are able to derive a few fundamental observations regarding malicious activities and their passive footprint:

- *Professionally-operated DNS zones have a unique passive network signal.*
- *Human-driven vs. automata-driven network actions are observable and differ, both at local and a global network level.*

These observations should be used in the next line of research projects that would utilize the research presented in this thesis. These new projects will effectively aim to propagate the signal obtained from the DNS abuse context, to address abuse in other services that make use of DNS or share the hosting infrastructure also used by the DNS resolutions. In the next few paragraphs, I will try to outline the most promising research directions for extending the research presented in this thesis.

Abuse in Social and Advertisement Networks: Social networks are growing rapidly in popularity. This viral growth of users makes them a very lucrative attack target. Attacks using the social networking platform could prove the most effective way of malware distribution. However new the attack vector may be, it will still have to rely upon the existing Internet infrastructure. As we saw from the research presented in Chapter 3, we have the methods to characterize as reputable or not hosts that facilitate resolution in our networks. The same signal can be used as a basic building block, along with features extracted from the problem domain (i.e., Twitter spam abuse [103]), to efficiently address abuse in social networking sites.

The same analogy can be made for advertisement networks in the mobile space. Almost every web page and application designed for mobile devices is serving some form of advertisement. Attackers are already taking advantage of ads and they are using them as an effective way to spread malware to the end users (malvertising). As in the case of abuse in social networks, the malicious ad will have to rely upon an existing Internet infrastructure. DNS reputation should be able to assist domain specific features, obtained from properties of the known malicious ad networks, toward the automatic identification of new malicious ad networks.

Abuse in Cloud Based Environments: Not surprisingly "cloud based" infrastructures are being utilized in a variety of scenarios from single users or big companies. Due to the reliability and scalability that cloud based platforms provide to users, attackers also have begun making use of cloud networks. Handling abuse in cloud networks is extremely critical for the reputation of the cloud provider and the other users of the cloud service. For example, if a cloud BGP prefix start facilitating a significant volume of illicit operation, this will most certainly affect the legitimate users of the same cloud BGP prefix (i.e., problem with email delivery and service reputation etc.). Furthermore, beside bad reputation, the cloud operators may have to deal

with extensive bandwidth costs if the malicious hosts are used for certain categories of fraudulent activities that require substantial bandwidth usage (i.e., spamming, malware distribution, etc.).

Since the abuse in a cloud environment needs to be dealt with in a proactive manner and at the edge of the cloud environment, a new generation of detection systems are required that will be able to model the patterns which Internet hosts are using to access the cloud services. As we saw in Chapter 4, the properties of the malware generated access patterns significantly differs from the human generated ones. As in the case of DNS, malware-related abuse in the cloud environments could be distinguishable by the access properties of Internet hosts observed over temporal windows.

Network Attribution: To this date, network level attribution is an unsolved problem in network security. The problem of network attribution is quite complex since the attribution methodology needs to take under consideration a variety of different signals regarding the reputation of a network. Among others, such signals should include; malware volumes reaching into the network (i.e., C&Cs), the duration the malicious hosts are active in the network, the different types of malicious activities the network facilitates, the BGP/passive DNS connections with other fraudulent networks and the average DNS reputation on the domain names facilitated by the hosts in the network. This research would be extremely valuable both to the operational community and to law enforcement for better coordinating actions against criminal organizations that facilitate consistency illicit operations.

Basic Security Metrics for Network Security: As we saw already in this thesis, by passively observing DNS we can systematically associate domain names' zone structures with the IP address space. However, if we aim to generically describe

network level events, new research is necessary that will be able to quantify network-protocol agnostic security measures. In both Chapters 4 and 5, we observed two main things about network access events. Firstly, DNS resolution patterns could be used to differentiate between benign and malicious behaviors. Secondly, spectral clustering could be used to identify groups of hosts that have similarities in their resolution requests (i.e., NXDomain resolution request patterns). To that extent we could leverage DNS reputation and access patterns to quantify basic metrics around the "security entropy" of the communication observed in networks. It appears that DNS — as a core infrastructure protocol for the Internet — could be a good first building block for devising basic metrics that could measure the security state of network devices.

6.4 Closing Remarks

This thesis explored the use of machine learning on passive DNS observation in order to devise systems that can preemptively detect DNS abuse. The three key contributions are the architecture of the first dynamic reputation system for DNS, and an early warning system for DNS — divided into two components. The first component enables DNS operators to preemptively detect malware-related domain names at the upper layers of the DNS hierarchy — in the absence of a related (to the domain name) malware sample. The second component of the early warning system enables DNS operators to preemptively detect the rise of DGA-botnets at the local recursive DNS or at the edge of the network — again in the absence of a related (to the domain name) malware sample and the corresponding DGA algorithm.

The systems presented in this thesis solve the DNS reputation problem at a basic level. The next generation of research should use the systems and classification signals introduced with this thesis, in order to address Internet abuse in other contexts (as described in Section 6.3), as long as the Internet abuse can be projected back to the

core Internet infrastructure that the DNS protocol also uses.

APPENDIX A

ANCILLARY INFORMATION

A.1 NXDomain Graph Analysis

As discussed in Section 5.3.1.3 we create a bipartite graph between NXDomains and hosts. Our interest is in the relationships between the NXDomains in the bipartite graph. Therefore, we have to convert the bipartite graph in a regular graph between NXDomains.

Spectral clustering requires the computation of the first p eigenvectors of S and then X-means clustering on them (see Algorithm 1). However, we do not need to compute S directly. The complexity for computing $S = M^T \cdot M$ is quadratic with respect to the number NXDomains, which will be prohibitive for scaling with very large datasets (i.e., across multiple DNS sensors in the same ISP). The eigenvalues of S are related to the singular $\sigma(M)$ values of S, according to:

$$\lambda(S) = \sqrt{\sigma(M)}$$

Also the left singular vectors of M are the same with the eigenvectors of S. Making use of this property is important since it avoids the $O(n^2)$ complexity step, without diluting the quality of the derived clusters.

A.2 Probabilistic Analysis of the DGA process

The botmaster that crafts a DGA malware can control three parameters that directly affect the life cycle of the DGA-enabled malware: (a) the number of generated domain names N by the DGA, (b) the number of domain names queried by the malware per epoch (i.e., one day) k, (c) the number of domain names q that the botmaster registers every day as C&Cs, in order to retain control of the botnet. Given the three

parameters (N, k, q) the probability $P(N, k, q)$ of a domain name resolution request x actually resolves to a registered (non-NXDomain) C&C can be computed as:

$$P(N, k, q) = 1 - \prod_{0}^{k-1}(1 - \frac{q}{N - i}) \quad (1)$$

In Figures 38, 39 and 40, we can observe the effect of every parameter with the respect of the others. It is logical to assume that the botmaster wants to register very few domain names every day because of the cost. Now, given a constant number of generated domain names N and a constant number of requests k we observe in Figure 38 that decreasing q will also decrease the probability of contacting the active C&C domain name. However, for constant values of k and q, decreasing N will increase the probability of contacting one of the active C&C domains, but as we see in Figure 39 (and we elaborate further in Section A.2.2) it also makes the behavior of the DNS request pattern less random and more predictable. Finally, for constant values of q and N, increasing k, also increases the probability of contacting the active C&C (Figure 40), but at the same time increases the DNS resolution activity for the infected hosts — which inevitably will also cause more unsuccessful resolutions.

A.2.1 Selecting the parameters of the DGA classifier

In Section 5.4.2 we discussed the hosts classification properties of the DGA classifier. We compile vectors using set of NXDomains with cardinality $\alpha = 10$ NXDomains. Given the probabilistic analysis of the Section A.2 we can further motivate our choice of using sets of 10 NXDomains. For known DGA's like Conficker-C the parameters (N, k, q) are known. Conficker-C generates 1000 domain names per day. In Figure 40 we see the probability for a host infected by Conficker-C of reaching a hypothetical active C&C domain. In particular, if we set $k = 10$, the probability of reaching the active C&C within the first 10 resolution attempt is only 0.01%.

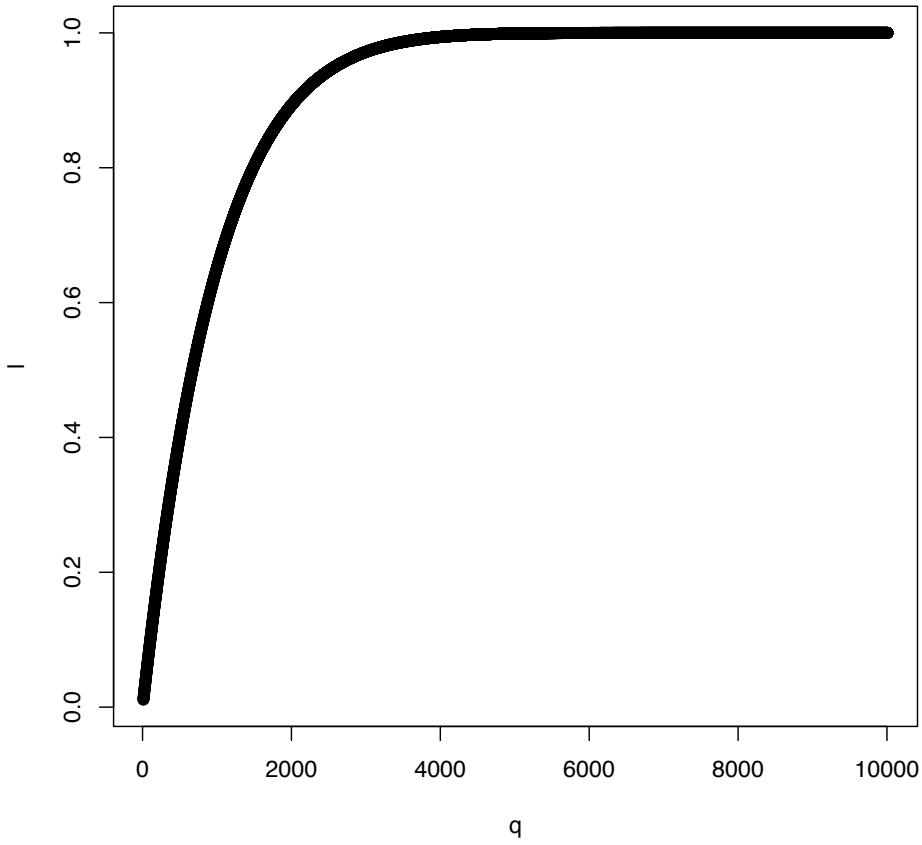


Figure 38: Using $N = 1000$, $k = 10$ and q varying.

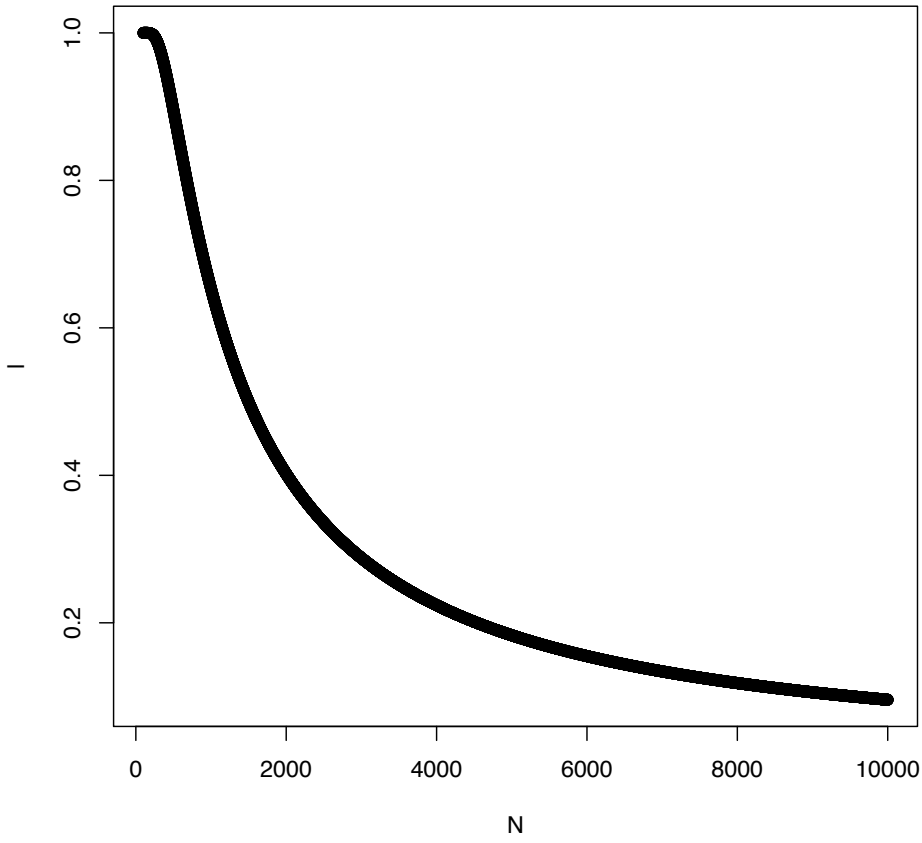


Figure 39: Using $q = 100$, $k = 10$ and N varying.

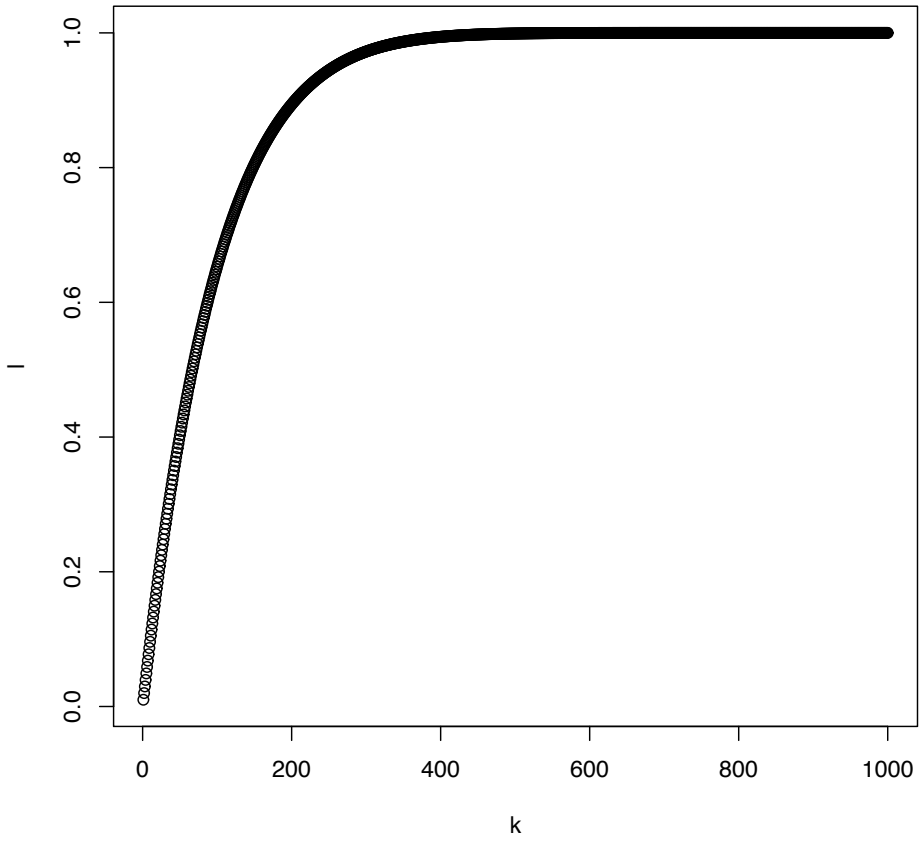


Figure 40: Using $N = 1000$, $q = 10$ and k varying.

A.2.2 The effect of DGA parameters on the graph clustering performance

In this section we investigate the effect of the three parameters (N, k, q) in the graph clustering module (Section 5.3). We will show that if botmasters intend to evade Pleiades graph clustering process, they will have to either increase the operating cost of their botnet infrastructure or significantly raise the number of NXDomains they generate every day — making the botnet DGA easier to track. Lets assume that a DGA-based malware generates $N \sim 10^4$ domain names but only queries k (i.e., Conficker-C). In this case, the probability of a domain name being queried by an infected host is $p(nx) \sim \frac{k}{N}$. Because the query events on two hosts infected by the same DGA are independent, the probability that the NXDomain is queried by l different hosts is $p_k(nx) = \prod_{i=1}^{l} p_i(nx) = \left(\frac{k}{N}\right)^l$. The same is the probability that l hosts that belong to the same botnet form a connected graph.

This probability p_k monotonically decreases with respect to k. So the adversary wants to keep k as low as possible to break down the connectivity of the graph. As we have already seen in Section A.2, in order to increase the probability of an infected host successful reaching the active C&C domain name, the adversary has to increase either m or k. Increasing m is costly, since the adversary will have to register a significant number of domain names per day. However, by increasing k, the cost is less, but comes with the cost of creating a significant higher number of NXDomains. Such an act from the botmaster will increase the connectivity of the NXDomain bipartite graph and thus the quality of our clusters.

REFERENCES

[1] Aas, K. and Eikvil, L., "Text categorisation: A survey.," 1999.

[2] Abu Rajab, M., Zarfoss, J., Monrose, F., and Terzis, A., "A multifaceted approach to understanding the botnet phenomenon," in *Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, IMC '06, 2006.

[3] abuse.ch, "ZeuS Gets More Sophisticated Using P2P Techniques." `http://www.abuse.ch/?p=3499`, 2011.

[4] Aha, D. W., Kibler, D., and Albert, M. K., "Instance-based learning algorithms," *Machine Learning*, vol. 6, pp. 37–66, 1991. 10.1023/A:1022689900470.

[5] Alexa, "The web information company." `http://www.alexa.com/`, 2007.

[6] Anderson, D., Fleizach, C., Savage, S., and Voelker, G., "Spamscatter: Characterizing internet scam hosting infrastructure," in *Proceedings of the USENIX Security Symposium*, 2007.

[7] Antonakakis, M., Perdisci, R., Dagon, D., Lee, W., and Feamster, N., "Building a dynamic reputation system for DNS," in *the Proceedings of 19th USENIX Security Symposium (USENIX Security '10)*, 2010.

[8] Antonakakis, M., Perdisci, R., Lee, W., Dagon, D., and Vasiloglou, N., "Detecting Malware Domains at the Upper DNS Hierarchy," in *the Proceedings of 20th USENIX Security Symposium (USENIX Security '11)*, 2011.

[9] Antonakakis, M., Perdisci, R., Nadji, Y., Vasiloglou, N., Abu-Nimeh, S., Lee, W., and Dagon, D., "From Throw-Away Traffic to Bots: Detecting the Rise of DGA-Based Malware," in *the Proceedings of 21th USENIX Security Symposium (USENIX Security '12)*, 2012.

[10] BankPatch, "Trojan.Bankpatch.C." `http://www.symantec.com/security_response/writeup.jsp?docid=2008-081817-1808-99&tabid=2`, 2009.

[11] Bilge, L., Kirda, E., Kruegel, C., and Balduzzi, M., "Exposure: Finding malicious domains using passive dns analysis," in *Proceedings of NDSS*, 2011.

[12] Bishop, C. M., *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2006.

[13] Breiman, L., "Bagging predictors," *Machine learning*, vol. 24, no. 2, pp. 123–140, 1996.

[14] Caballero, J., Grier, C., Kreibich, C., and Paxson, V., "Measuring pay-per-install: The commoditization of malware distribution," in *USENIX Security Symposium*, 2011.

[15] Chang, C.-C. and Lin, C.-J., "LIBSVM: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, pp. 27:1–27:27, 2011. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[16] Collins, M. P., Shimeall, T. J., Faber, S., Janies, J., Weaver, R., De Shon, M., and Kadane, J., "Using uncleanliness to predict future botnet addresses," in *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, IMC '07, (New York, NY, USA), pp. 93–104, ACM, 2007.

[17] Consortium, I. S., "SIE@ISC : Security Information Exchange." `https://sie.isc.org/`, 2004.

[18] Cova, M., Leita, C., Thonnard, O., Keromytis, A. D., and Dacier, M., "An analysis of rogue av campaigns," in *RAID*, pp. 442–463, 2010.

[19] Dagon, D., Gu, G., and Lee, C. P., "A taxonomy of botnet structures," in *Botnet Detection*, pp. 143–164, 2008.

[20] Dagon, D., Zou, C., and Lee, W., "Modeling botnet propagation using time zones," in *In Proceedings of the 13 th Network and Distributed System Security Symposium NDSS*, 2006.

[21] dihe's IP-Index Browser, "DIHE." `http://ipindex.homelinux.net/index.php`, 2008.

[22] Dinaburg, A., Royal, R., Sharif, M., and Lee, W., "Ether: malware analysis via hardware virtualization extensions," in *ACM CCS*, 2008.

[23] DNS Whitelist – Protect against false positives, "DNSWL." `http://www.dnswl.org`, 2008.

[24] DNSBL, S., "Fighting spam by finding and listing Exploitable Servers.." `http://www.us.sorbs.net/`, 2007.

[25] Duda, R., Hart, P., and Stork, D., *Pattern Classification*. Wiley-Interscience, 2nd ed., 2000.

[26] Feldman, R. and Sanger, J., *The text mining handbook: advanced approaches in analyzing unstructured data*. Cambridge Univ Pr, 2007.

[27] Felegyhazi, M., Keibich, C., and Paxson, V., "On the potential of proactive domain blacklisting," in *Third USENIX LEET Workshop*, 2010.

[28] Ferguson, P. and Senie, D., "Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing (BCP38)." http://www.ietf.org/rfc/bcp/bcp38.txt, 2000.

[29] Ferguson, P., "Network ingress filtering: Defeating denial of service attacks which employ ip source address spoofing extension mechanisms for dns (edns0), rfc 2671," 2000.

[30] Finones, R., "Virus:Win32/Expiro.Z." http://www.microsoft.com/security/portal/Threat/Encyclopedia/Entry.aspx?Name=Virus%3AWin32%2FExpiro.Z, 2011.

[31] Fogla, P., Sharif, M., Perdisci, R., Kolesnikov, O., and Lee, W., "Polymorphic blending attacks," in *In Proceedings of the 15 th USENIX Security Symposium*, pp. 241–256, 2006.

[32] Freund, Y. and Mason, L., "The alternating decision tree learning algorithm," in *Proceedings of the Sixteenth International Conference on Machine Learning*, ICML '99, 1999.

[33] G. Gu, P. Porras, V. Yegneswaran, M. Fong, and W. Lee, "BotHunter: Detecting malware infection through IDS-driven dialog correlation," in *Proc. USENIX Security*, 2007.

[34] Garera, S., Provos, N., Chew, M., and Rubin, A., "A framework for detection and measurement of phishing attacks," in *Proceedings of the ACM WORM*, ACM, 2007.

[35] Geide, M., "Another trojan bamital pattern." http://research.zscaler.com/2011/05/another-trojan-bamital-pattern.html, 2011.

[36] Golovanov, S. and Soumenkov, I., "TDL4 top bot." http://www.securelist.com/en/analysis/204792180/TDL4_Top_Bot, 2011.

[37] Gross, B., Cova, M., Cavallaro, L., Gilbert, B., Szydlowski, M., Kemmerer, R., Kruegel, C., and Vigna, G., "Your botnet is my botnet: analysis of a botnet takeover," in *ACM CCS 09*, (New York, NY, USA), ACM, 2009.

[38] Gu, G., Perdisci, R., Zhang, J., and Lee, W., "BotMiner: clustering analysis of network traffic for protocol- and structure-independent botnet detection," in *USENIX Security*, 2008.

[39] Gu, G., Zhang, J., and Lee, W., "BotSniffer: Detecting botnet command and control channels in network traffic," in *Network and Distributed System Security Symposium (NDSS)*, 2008.

[40] Hao, S., Feamster, N., and Pandrangi, R., "An Internet Wide View into DNS Lookup Patterns." http://labs.verisign.com/projects/malicious-domain-names/white-paper/dns-imc2010.pdf, 2010.

[41] Hastie, T., Tibshirani, R., and Friedman, J., *The Elements of Statistical Learning*. Springer Series in Statistics, New York, NY, USA: Springer New York Inc., 2001.

[42] Hermans, J., "MozillaWiki TLD List." https://wiki.mozilla.org/TLD_List, 2006.

[43] Holz, T., Gorecki, C., Rieck, K., and Freiling, F., "Measuring and detecting fast-flux service networks," in *Proceedings of NDSS*, 2008.

[44] Hothorn, T. and Lausen, B., "Double-bagging: Combining classifiers by bootstrap aggregation," *Pattern Recognition*, vol. 36, no. 6, pp. 1303–1309, 2003.

[45] John, G. and Langley, P., "Estimating continuous distributions in bayesian classifiers," in *In Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*, pp. 338–345, Morgan Kaufmann, 1995.

[46] Jung, J., Krishnamurthy, B., and Rabinovich, M., "Flash crowds and denial of service attacks: characterization and implications for cdns and web sites," in *Proceedings of the 11th international conference on World Wide Web*, WWW '02, (New York, NY, USA), pp. 293–304, ACM, 2002.

[47] Jung, J., Sit, E., Balakrishnan, H., and Morris, R., "DNS performance and the effectiveness of caching," *IEEE/ACM Trans. Netw.*, vol. 10, pp. 589–603, October 2002.

[48] Kanich, C., Weaver, N., McCoy, D., Halvorson, T., Kreibich, C., Levchenko, K., Paxson, V., Voelker, G. M., and Savage, S., "Show me the money: Characterizing spam-advertised revenue," in *USENIX Security Symposium*, 2011.

[49] Krishnan, S. and Monrose, F., "Dns prefetching and its privacy implications: when good things go bad," in *Proceedings of the 3rd USENIX conference on Large-scale exploits and emergent threats: botnets, spyware, worms, and more*, LEET'10, (Berkeley, CA, USA), pp. 10–10, USENIX Association, 2010.

[50] Kuncheva, L. I., *Combining Pattern Classifiers: Methods and Algorithms*. Wiley-Interscience, 2004.

[51] Levchenko, K., Pitsillidis, A., Chachra, N., Enright, B., Félegyházi, M., Grier, C., Halvorson, T., Kanich, C., Kreibich, C., Liu, H., McCoy, D., Weaver, N., Paxson, V., Voelker, G. M., and Savage, S., "Click trajectories: End-to-end analysis of the spam value chain," in *IEEE Symposium on Security and Privacy*, pp. 431–446, 2011.

[52] Ligh, M. H., Adair, S., Hartstein, B., and Richard, M., *Malware Analyst's Cookbook and DVD*, ch. 12. Wiley, 2010.

[53] Lu, L., Perdisci, R., and Lee, W., "Surf: detecting and measuring search poisoning," in *Proceedings of the 18th ACM conference on Computer and communications security*, CCS '11, (New York, NY, USA), ACM, 2011.

[54] MalwareDomains, "DNS-BH malware domain blocklist." `http://www.malwaredomains.com`, 2007.

[55] Mockapetris, P., "Domain names - concepts and facilities." `http://www.ietf.org/rfc/rfc1034.txt`, 1987.

[56] Mockapetris, P., "Domain names - implementation and specification." `http://www.ietf.org/rfc/rfc1035.txt`, 1987.

[57] Newman, M., *Networks: an introduction*. Oxford University Press, 2010.

[58] Ng, A. Y., Jordan, M. I., and Weiss, Y., "On spectral clustering: Analysis and an algorithm," in *Advances In Neural Information Processing Systems*, pp. 849–856, MIT Press, 2001.

[59] OPENDNS, "OpenDNS — Internet Navigation And Security." `http://www.opendns.com/`, 2010.

[60] P. Porras, H. Saidi, and V. Yegneswaran, "An Analysis of Conficker's Logic and Rendezvous Points." `http://mtc.sri.com/Conficker/`, 2009.

[61] Pelleg, D. and Moore, A. W., "X-means: Extending k-means with efficient estimation of the number of clusters," in *Proceedings of the Seventeenth International Conference on Machine Learning*, ICML '00, (San Francisco, CA, USA), pp. 727–734, Morgan Kaufmann Publishers Inc., 2000.

[62] Perdisci, R., Lee, W., and Feamster, N., "Behavioral clustering of HTTP-based malware and signature generation using malicious network traces," in *USENIX NSDI*, 2010.

[63] Plonka, D. and Barford, P., "Context-aware clustering of DNS query traffic," in *Proceedings of the 8th IMC*, (Vouliagmeni, Greece), ACM, 2008.

[64] POLSKA, C., "ZeuS P2P+DGA variant mapping out and understanding the threat." `http://www.cert.pl/news/4711/langswitch_lang/en`, 2012.

[65] Porras, P., "Inside risks: Reflections on conficker," *Communications of the ACM*, vol. 52, pp. 23–24, October 2009.

[66] Porras, P., Saidi, H., and Yegneswaran, V., "Conficker C analysis," tech. rep., SRI International, Menlo Park, CA, April 2009.

[67] Project, T. S., "ZEN - Spamhaus DNSBLs." http://www.spamhaus.org/zen/, 2004.

[68] Provos, N., Mavrommatis, P., Rajab, M. A., and Monrose, F., "All your iframes point to us," in *USENIX Security Symposium*, pp. 1–16, 2008.

[69] Provos, N., McNamee, D., Mavrommatis, P., Wang, K., and Modadugu, N., "The Ghost in the Browser: Analysis of Web-based Malware," in *Usenix Hotbots 2007*, April 2007.

[70] R. Perdisci, I. Corona, D. Dagon, and W. Lee, "Detecting malicious flux service networks through passive analysis of recursive DNS traces," in *Proceedings of ACSAC*, (Honolulu, Hawaii, USA), 2009.

[71] Rabiner, L. R., "Readings in speech recognition," ch. A tutorial on hidden Markov models and selected applications in speech recognition, 1990.

[72] Royal, P., "Analysis of the kraken botnet." http://www.damballa.com/downloads/r_pubs/KrakenWhitepaper.pdf, 2008.

[73] S. Hao, N. Syed, N. Feamster, A. Gray and S. Krasser, "Detecting spammers with SNARE: Spatio-temporal network-level automatic reputation engine," in *Proceedings of the USENIX Security Symposium*, 2009.

[74] S. Shevchenko, "Srizbi Domain Generator Calculator." http://blog.threatexpert.com/2008/11/srizbis-domain-calculator.html, 2008.

[75] Sato, K., Ishibashi, K., Toyono, T., and Miyake, N., "Extending black domain name list by using co-occurrence relation between dns queries," in *Third USENIX LEET Workshop*, 2010.

[76] Sculley, D., Otey, M. E., Pohl, M., Spitznagel, B., Hainsworth, J., and Zhou, Y., "Detecting adversarial advertisements in the wild," in *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '11, ACM, 2011.

[77] Shevchenko, S., "Domain Name Generator for Murofet." http://blog.threatexpert.com/2010/10/domain-name-generator-for-murofet.html, 2010.

[78] Sinha, S., Bailey, M., and Jahanian, F., "Shades of grey: On the effectiveness of reputation-based blacklists," in *3rd International Conference on MALWARE*, 2008.

[79] Snow, K. Z., Krishnan, S., Monrose, F., and Provos, N., "Shellos: Enabling fast detection and forensic analysis of code injection attacks," in *USENIX Security Symposium*, 2011.

[80] SOPHOS, "Mal/Simda-C." http://www.sophos.com/en-us/threat-center/threat-analyses/viruses-and-spyware/Mal~Simda-C/detailed-analysis.aspx, 2012.

[81] Staniford, S., Paxson, V., and Weaver, N., "How to own the internet in your spare time," in *Proceedings of the 11th USENIX Security Symposium*, (Berkeley, CA, USA), pp. 149–167, USENIX Association, 2002.

[82] Statistics, L. B. and Breiman, L., "Random forests," in *Machine Learning*, pp. 5–32, 2001.

[83] Stewart, J., "Bobax trojan analysis." http://www.secureworks.com/research/threats/bobax/, 2004.

[84] Stone-Gross, B., Cova, M., Cavallaro, L., Gilbert, B., Szydlowski, M., Kemmerer, R., Kruegel, C., and Vigna, G., "Your botnet is my botnet: analysis of a botnet takeover," in *Proceedings of the 16th ACM conference on Computer and communications security*, CCS '09, (New York, NY, USA), pp. 635–647, ACM, 2009.

[85] Stone-Gross, B., Cova, M., Kruegel, C., and Vigna, G., "Peering through the iframe," in *INFOCOM*, pp. 411–415, 2011.

[86] Stover, S., Dittrich, D., Hernandez, J., and Dietrich, S., "Analysis of the storm and nugache trojans: P2P is here," in *USENIX ;login:, vol. 32, no. 6*, December 2007.

[87] T.-F. Yen and M. K. Reiter, "Are your hosts trading or plotting? Telling P2P file-sharing and bots apart," in *ICDCS*, 2010.

[88] Team Cymru, "IP to ASN mapping." http://www.team-cymru.org/Services/ip-to-asn.html, 2008.

[89] The Conficker Working Group, "Conficker working group: Lessons learned." 2011.

[90] The Honeynet Project & Research Alliance, "Know Your Enemy: Fast-Flux Service Networks." http://old.honeynet.org/papers/ff/fast-flux.html, 2007.

[91] URIBL, "Real time URI blacklist." http://uribl.com.

[92] Villamarin-Salomon, R. and Brustoloni, J., "Identifying botnets using anomaly detection techniques applied to dns traffic," in *5th Consumer Communications and Networking Conference*, 2008.

[93] Vixie, P., "DNS complexity," *ACM Queue*, vol. Vol. 5, no. Num. 3, pp. pages: 24–29, 2007.

[94] Weaver, N., Staniford, S., and Paxson, V., "Very fast containment of scanning worms," in *In Proceedings of the 13th USENIX Security Symposium*, pp. 29–44, 2004.

[95] Weimer, F., "Passive DNS replication," in *Proceedings of FIRST Conference on Computer Security Incident*, (Hand ling, Singapore), 2005.

[96] Wessels, D., Fomenkov, M., Brownlee, N., and Claffy, K., "Measurements and laboratory simulations of the upper DNS hierarchy," in *PAM*, 2004.

[97] Wikipedia, "The storm botnet." `http://en.wikipedia.org/wiki/Storm_botnet`, 2010.

[98] Williams, J., "What we know (and learned) from the waledac takedown." `http://tinyurl.com/7apnn9b`, 2010.

[99] Wolf, J., "Technical details of srizbi's domain generation algorithm." http://blog.fireeye.com/research/2008/11/technical-details-of-srizbis-domain-generation-algorithm.html, 2008. Retreived: April, 10 2010.

[100] Wong, J., "Trojan:Java/Boonana." `http://www.microsoft.com/security/portal/Threat/Encyclopedia/Entry.aspx?Name=Trojan%3AJava%2FBoonana`, 2011.

[101] Yadav, S. and Reddy, A. N., "Winning with dns failures: Strategies for faster botnet detection," in *7th International ICST Conference on Security and Privacy in Communication Networks*, 2011.

[102] Yadav, S., Reddy, A. K. K., Reddy, A. N., and Ranjan, S., "Detecting algorithmically generated malicious domain names," in *Proceedings of the 10th annual Conference on Internet Measurement*, IMC '10, (New York, NY, USA), pp. 48–61, ACM, 2010.

[103] Yang, C., Harkreader, R. C., and Gu, G., "Die free or live hard? empirical evaluation and new design for fighting evolving twitter spammers," in *RAID*, pp. 318–337, 2011.

[104] Yen, T.-F. and Reiter, M. K., "Traffic aggregation for malware detection," in *Proc. International conference on Detection of Intrusions and Malware, and Vulnerability Assessment (DIMVA)*, 2008.

[105] Z. Qian, Z. Mao, Y. Xie and F. Yu, "On network-level clusters for spam detection," in *Proceedings of the USENIX NDSS Symposium*, 2010.

[106] Zdrnja, B., Brownlee, N., and Wessels, D., "Passive monitoring of DNS anomalies," in *Proceedings of DIMVA Conference*, 2007.

[107] Zdrnja, B., "Google Chrome and (weird) DNS requests." http://isc.sans.edu/diary/Google+Chrome+and+weird+DNS+requests/10312, 2011.

[108] Zeus Tracker, "Zeus IP & domain name block list." https://zeustracker.abuse.ch, 2009.

[109] Zhang, J., Porra, P., and Ullrich, J., "Highly predictive blacklisting," in *Proceedings of the USENIX Security Symposium*, 2008.

[110] Zhang, J., Perdisci, R., Lee, W., Sarfraz, U., and Luo, X., "Detecting stealthy P2P botnets using statistical traffic fingerprints," in *Annual IEEE/IFIP International Conference on Dependable Systems and Networks - Dependable Computing and Communication Symposium*, 2011.